Mälardalen University
School of Innovation Design and Engineering
Västerås, Sweden

Thesis for the Degree of Master of Science in Engineering - Dependable Systems 30.0 credits

DATASET GENERATION IN A SIMULATED ENVIRONMENT USING REAL FLIGHT DATA FOR RELIABLE RUNWAY DETECTION CAPABILITIES

Emil Tagebrand
etagebrand@hotmail.com

Emil Gustafsson Ek
emil_ege@live.se

Examiner: Masoud Daneshtalab
Mälardalen University, Västerås, Sweden

Supervisor(s): Håkan Forsberg
Mälardalen University, Västerås, Sweden

Company Supervisor(s): Joakim Lindén
SAAB, Järfälla, Sweden

21/06/2021

Acknowledgements

We want to thank SAAB for giving us the opportunity in collaborating on this master thesis. From SAAB, we also want to thank both Joakim Lindén and Erasmus Cedernaes for providing guidance, resources and equipment for executing this work. Furthermore, we also would like to express gratitude to Dr Håkan Forsberg and Johan Hjorth at Mälardalen University for both aid and guidance in the report and for showing enthusiasm for the research. Lastly, we would like to thank Josef Haddad for enlightening us in applying certain methods and giving aid in the training of algorithms.

Abstract

Implementing object detection methods for runway detection during landing approaches is limited in the safety-critical aircraft domain. This limitation is due to the difficulty that comes with verification of the design and the ability to understand how the object detection behaves during operation. During operation, object detection needs to consider the aircraft's position, environmental factors, different runways and aircraft attitudes. Training such an object detection model requires a comprehensive dataset that defines the features mentioned above. The feature's impact on the detection capabilities needs to be analysed to ensure the correct distribution of images in the dataset. Gathering images for these scenarios would be costly and needed due to the aviation industry's safety standards. Synthetic data can be used to limit the cost and time required to create a dataset where all features occur. By using synthesised data in the form of generating datasets in a simulated environment, these features could be applied to the dataset directly. The features could also be implemented separately in different datasets and compared to each other to analyse their impact on the object detections capabilities. By utilising this method for the features mentioned above, the following results could be determined. For object detection to consider most landing cases and different runways, the dataset needs to replicate real flight data and generate additional extreme landing cases. The dataset also needs to consider landings at different altitudes, which can differ at a different airport. Environmental conditions such as clouds and time of day reduce detection capabilities far from the runway, while attitude and runway appearance reduce it at close range. Runway appearance did also affect the runway at long ranges but only for darker runways.

Acronyms

ADS-B Automatic Dependent Surveillance–Broadcast.

ANN Artificial Neural Network.

AP Average Precision.

API Application Programming Interface.

APl Average Precision Large.

APm Average Precision Medium.

APs Average Precision Small.

ATC Air Traffic Control.

CNN Convolutional Neural Network.

COCO Common Objects in COntext.

CPF Correlated Position reports for a Flight.

DDR Demand Data Repository.

DNN Deep Neural Network.

EASA European Union Aviation Safety Agency.

FAA Federal Aviation Administration.

GNSS Global Navigation Satellite System.

ICAO International Civil Aviation Organization.

ILS Instrument Landing System.

IoU Intersection over Union.

mAP Mean Average Precision.

METAR METeorological Aerodrome Report.

ML Machine learning.

MS-COCO Microsoft Common Objects in COntext.

NN Neural Network.

PDF Probability Distribution Function.

ResNet Residual Neural Network.

ResNeXt Residual Neural Network Next Dimension.

SESAR Single European Sky ATM (Air Traffic Management) Research.

TMA Terminal Maneuvering Area.

UAV Unmanned Aerial Vehicle.

VFR Visual Flight Rules.

XPC X-Plane Connect.

Contents

List of Figures

List of Tables

1. Introduction

In the aviation domain, machine learning has generated impressive results in the field of vision. With advances in computational capabilities and access to large amounts of data, machine learning can be applied to object detection and image segmentation [1]. For the safety-critical domain implementing methods such as object detection and image segmentation is limited. This limitation is due to the difficulty that comes with verification of the design and the ability to understand how it behaves during operation [1].

A description made by European Union Aviation Safety Agency (EASA) [1, p. 7] for machine learning in aviation is as follow: "Machine learning therefore provides major opportunities for the aviation industry, yet the trustworthiness of such systems needs to be guaranteed".

This paper aims to understand how an object detection model is affected by particular variation in images, specifically images of runways during a landing approach. Several methods are applied to generate both realistic images of an aircraft's perspective during landing and facilitate an environment that enables applying several types of variations. These variations being changing airport, alternating environmental conditions and the attitude of the aircraft. Several methods of replicating distributions from real flight data are also analysed. These methods are necessary to ensure that most possible cases of landings are trained on, even those occurring rarely.

In this paper, an image set for training neural networks comes from synthetic images. The synthetic imagery is produced from a simulated environment where the images represent what an aircraft observes during a landing approach. This view is necessary as runway detection is paramount. After training, the object detection model is produced. This model can then predict the location of the runway in a new set of images.

Real-world data of landing approaches aid in replicating realistic landing approach scenarios. Due to recent advances in flight systems, real-world aircraft information is more obtainable. Automatic Dependent Surveillance–Broadcast (ADS-B), allows for simple access of aircraft state information.

There can be large variations in the environment around a runway. These variations are due to the dissimilar geographical locations runways are present. Producing image sets by recording landing approaches in the real world is expensive. Particularly considering the recording plane would need to operate in many environmental conditions to create a comprehensive image set.

The end goal of this paper is to allow for generating reliable neural networks in the vision domain. Reliability is increased by understanding how a network responds to variations, thus knowing which variations are necessary to train on for increasing prediction accuracy. This may advance the application of vision-based neural networks in the safety-critical domain.

Section 1 contains the introduction, where the problem formulation and research questions are defined. The background to this thesis is located in section 2. Section 3 is the related work, and section 4 describes the methods used for this thesis. The ethical and societal considerations are defined in section 5. Results from analysis and experimentation are found in section 6. The discussion in section 7 analyses the results, and, lastly, the conclusion is found in section 8.

1.1 Problem Formulation

Visual-based object detection in conjunction with neural networks is investigated in multiple industries to improve system awareness of its surroundings. Implementation of such a system requires high accuracy and robustness to be deemed safe. A neural network only learns as much as the dataset provides knowledge about the environment. Thus the given dataset must describe the environment accurately and according to real scenarios. To create datasets with such accuracy requires a lot of data and the possibility to generate all possible scenarios that can occur. The dataset must also know all the crucial features in the environment. In the aviation industry, this would be costly but also needed due to the dependability standard in aviation. When considering datasets which train neural networks for runway detection capabilities when landing, additional complications arise. Landing approaches differ based on weather conditions, airports, and aircraft. Such differences affect the applicability of the system between different landing scenarios and airports. This thesis work focuses on generating synthetic datasets in a simulator to represent real landing scenarios. Manipulation of the simulated environment allows it to cover a wide range of

landing scenarios. However, differences such as the simulated environment's detail compared to the real world affect the accuracy and thus, the system's applicability. The simulator also needs to present the landing scenario accurately according to how it is performed in real life. Thus, the investigation will focus on analysing what features are important while creating a dataset for runway object detection.

1.2 Research Questions

Based on the problems and limitations mentioned in the problem formulation, the following research questions where derived.

- RQ1: What method of generating distributions from real data produces the highest prediction accuracy from generated images?
- RQ2: How does changing airport runways affect the prediction accuracy of the object detection?
- RQ3: How does environmental conditions affect prediction accuracy of the object detection?
- RQ4: How does aircraft attitude affect prediction accuracy of the object detection?
- RQ5: What variables need to be considered when generating reliable datasets used for safety-critical landing approaches?

2. Background

In this section, knowledge is presented, which is a prerequisite to understanding the procedures and results in this thesis.

2.1 Landing Approaches

To achieve a realistic dataset classifying landing approaches, all possible landing scenarios need to be covered. Based on the Federal Aviation Administration (FAA), factors such as wind speed, wind direction, turbulent wind, size of the runway, ground effects, aeroplane specifications, and obstacles affect how the aeroplane shall approach the runway. If landing conditions are not satisfactory for the approach, landing can be rejected. This rejection is usually due to unexpected hazards on the runway, wind turbulence or mechanical failures. In the case of a rejected landing, The aeroplane shall perform Go-Around where a new landing approach is performed under more favourable conditions [2].

2.1.1 Normal Landing Approach

The Standard landing process is considered when engine power is available and low wind is present. This process is divided into five different parts: the base leg, final approach, roundout, touchdown, and after-landing roll. From these parts, the base leg and final approach describe the approach sequence considered in this thesis and a simple example of their function can be seen in Figure 1. The base leg is performed by finding a gradually descending path resulting in a landing on a runway's designated point. To find this path; altitude, distance to runway, airspeed, and wind need to be considered. The base leg approach is used until a medium to shallow-banked turn can align the aeroplane towards the runway centerline. The final approach starts when the aeroplane is aligned towards the runway centerline. From this point; flaps settings and pitch shall be adjusted for the desired rate of descent and airspeed. The desired rate of descent shall correlate to a landing at the centre of the first third of the runway. Airspeed shall be maintained, so minimum floating is achieved before touchdown. Floating are sudden movements that affect the aircraft's trajectory when close to the runway [2].

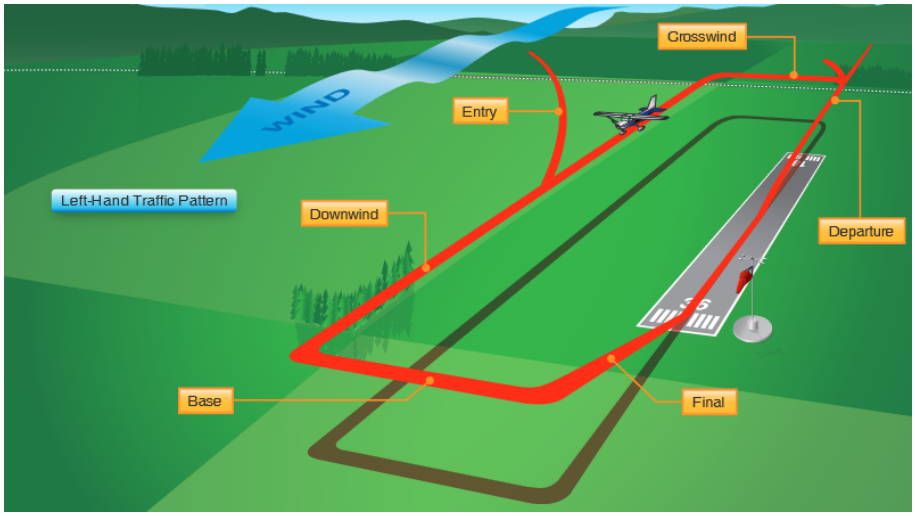


Figure 1: The figure is from FAA, Airplane Flying Handbook, Chapter 7 [3]. It shows a simplified view of the lifting and landing stages. Base leg is performed to find a gradually descending path resulting in a landing on a runway's designated point. The final approach focus on finding a desired rate of decent towards the runway.

2.1.2 Windy Conditions

Landing during windy conditions affects the aeroplane differently depending on the direction of the wind. Wind parallel to the runway, either facing or in the plane's direction, affects airspeed and lift. The pilot needs to consider these changes in airspeed and lift to land on the desired position. Wind orthogonal to the runway will cause the plane to drift towards the direction of the wind. During this wind condition, the crab method and the wing-low method can be used. The crab method focuses on angling the aeroplane towards the wind, which counteracts the drifting. The Wing-low method fulfils the same function as the crab method but instead alters the plane's roll [2].

2.2 Landing Configuration and Terminology

An imaginary line is drawn from the runway centerline that extends beyond the runway. This line has the same angle as the bearing of the runway. The imaginary line is useful as an aid in determining the lateral position of the aircraft relative to the runway. In this paper, the term "lateral position" denotes an aircraft's displacement around the runway centerline. The term is used throughout this paper and is visible as an example in Figure 2.

An aircraft coming in for a landing will be situated in close proximity to the runway centerline, as a larger deviation will prevent landing. A large deviation will require a larger correction which may have some risk.

2.2.1 Runway Appearance

The runway designator is based on the angle from the magnetic north to the centerline of the runway measured clockwise. In parallel runways, the runways will have an L or R for left or right, if three, L, C and R, where C is Centre. In the case of the runway in Figure 2, it should have a parallel runway to the right, as the designator is 24L. The opposite direction of the same runway will also have an identification with the opposite angle (180° shift) and letter. For example, in Figure 2, the opposite runway identification would be 06R [4].

There are several types of markings on runways to provide information to the pilots when performing interactions with the runway, such as landing or taking-off. Some of the markings are centerline, aimpoint, touchdown zone, threshold and threshold markings. Depending on the type of runway, some markings may not exist [4].

The width of the runway is determined by the number of stripes in the threshold markings. Table 1 shows the various widths of runways. The aimpoints purpose is to serve as a visual aid for landing. The markings on the runway are white [4]. If it is dark, the runway will look different, as there are lights that guides the pilot on approach. Some markings on the runway may not be visible.

Runway Width	Number of Stripes
60 feet (18 m)	4
75 feet (23 m)	6
100 feet (30 m)	8
150 feet (45 m)	12
200 feet (60 m)	16

Table 1: (Source: FAA Aeronautical Information Manual). Amount of lines in the threshold markings on runways and the corresponding width of the runway in feet.

2.2.2 Environment Around the Runway

There are also taxiing areas that may look similar to the runway. The runway markings are white. Other sections located around the runway are yellow. Some examples of sections where the markings are instead yellow are the taxiways, regions not intended for aircraft's and holding positions. There may be Air Traffic Control (ATC) towers present at airports, other aircraft's, maintenance vehicles, busses, and buildings. Many runways are situated close to cities, in which

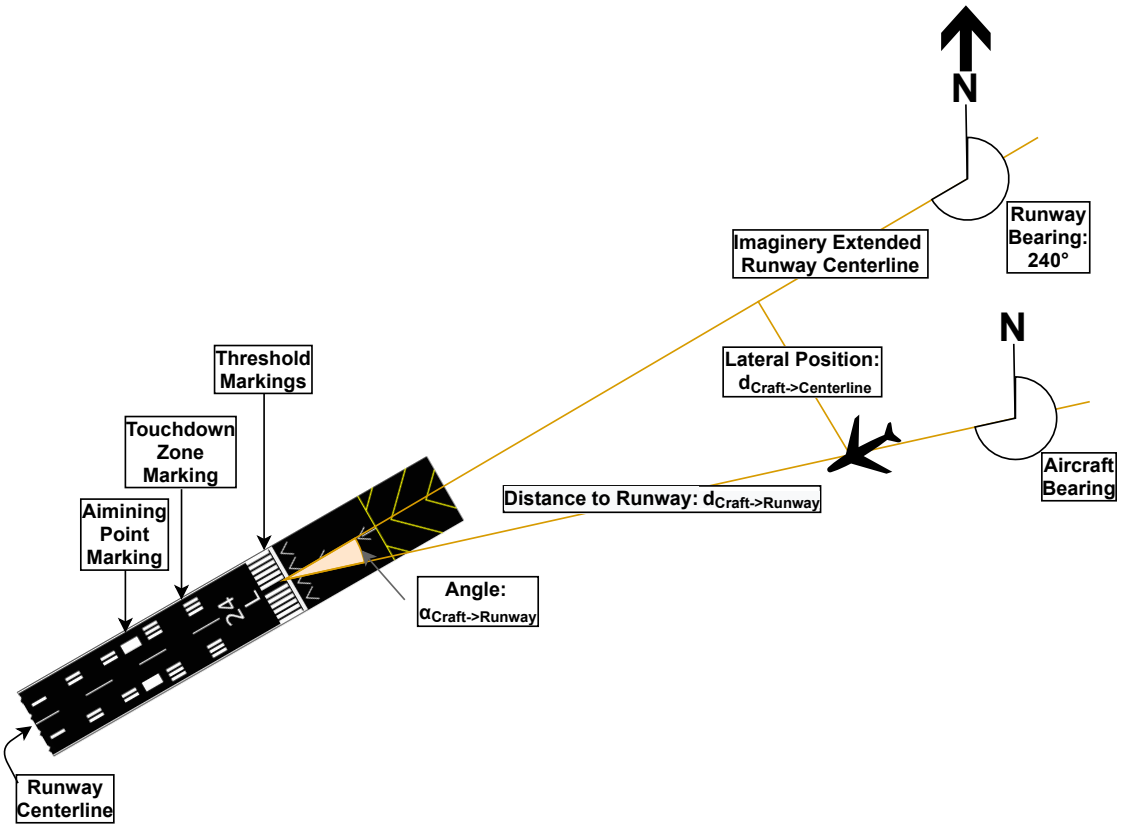


Figure 2: The runway in the image attribution: Mormegil, CC BY-SA 3.0, via Wikimedia Commons. Runway contains multiple markings such as aiming points to aid the pilot during landing. Terminology and setup of aircraft relative to runway ensure that correct landing procedure are performed. If correct setup is hard to sustain while landing, these variables can help identify a go around situation.

there are highways present that can look like runways. At larger airports, there may also be several runways present. They may be placed in parallel or crossing into each other such as the Skavsta Airport in Sweden.

2.3 Data

This section include both flight data from aircraft and information about the airports and runway locations. This data is both dynamic and static. Flight data is dynamic and can be received through a system called Automatic Dependent Surveillance–Broadcast (ADS-B). Location information about airports is readily available through the Internet and is static.

2.3.1 Automatic Dependent Surveillance–Broadcast

ADS-B is a system on aircraft or surface vehicles that broadcasts state vector or other information. The ADS-B system is a one-way communication system as it does not require any acknowledgement from a recipient. There exist two types of ADS-B. One type is called ADS-B OUT, which broadcast information from the aircraft, and the second type is called ADS-B IN [5]. Aircraft's with ADS-B IN installed can receive broadcasts from air traffic, flight and weather information [6]. ADS-B does not contain an aircraft's orientational information. Such as the roll, pitch or yaw of the aircraft.

An ADS-B OUT function will collect several parameters for broadcasting. The broadcasted state vector information contains a summary of tracking parameters such as position, velocity, track, and identity [7]. The position represents several parameters such as latitude, longitude and altitude. The positional information from ADS-B comes from Global Navigation Satellite System

(GNSS) [5]. Altitude contains information on barometric altitude and geometric altitude. The identity includes both an ICAO address of the ADS-B transponder and a callsign of the vehicle it is on. Besides the state vector information, there are also parameters such as time or squawk [7].

ADS-B complements the older radar technology, which only provides a refresh rate of 5-12 seconds, compared to ADS-B OUT, which sends a message every second. The ADS-B system sends their data to ground stations that allow air traffic controllers to know the aircraft's information. According to the FAA [6], there are approximately 700 ADS-B ground stations in the US.

As of January 2020, it is mandatory to have ADS-B OUT instrumentation when flying in most controlled airspace in the US [6]. ADS-B is a key part of Next Generation Air Transportation System (NextGen), an FAA-led modernisation of the US air transportation system that aids in making flying safer and more efficient [8]. Similar to the FAAs NextGen initiative, the Eurocontrol Single European Sky ATM (Air Traffic Management) Research (SESAR) aims at implementing ADS-B as a goal of modernisation [5].

Glitches and other issues are present in ADS-B data, as described in [9], [10]. Furthermore, an example of such an issue is visible in Figure 3, albeit not a major one. Altitude information from some aircraft contain much glitching, as visible in Figure 4.

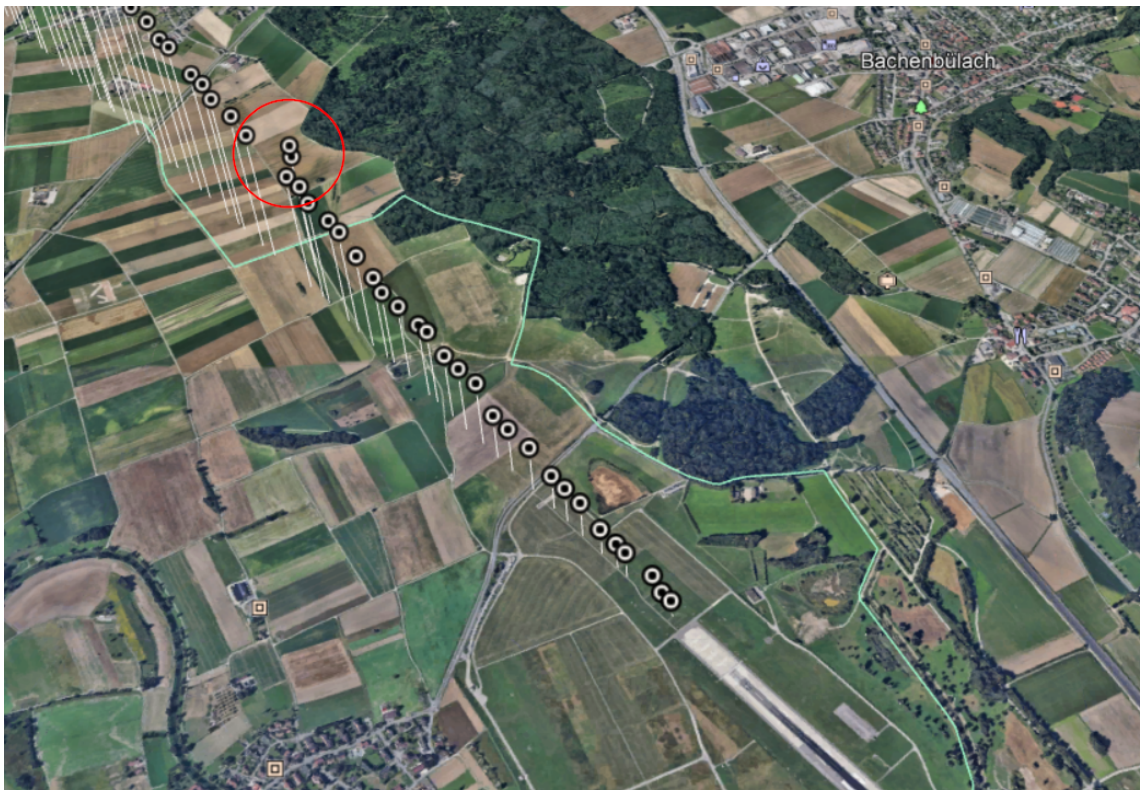


Figure 3: An example of a flight trajectory when landing. The white dots represent each ADS-B message received from the aircraft. This landing takes place on runway 14 at Zürich airport (LSZH) in Switzerland. The red circle indicates ADS-B messages, which have an error, as such a deviation is not possible on the aircraft.

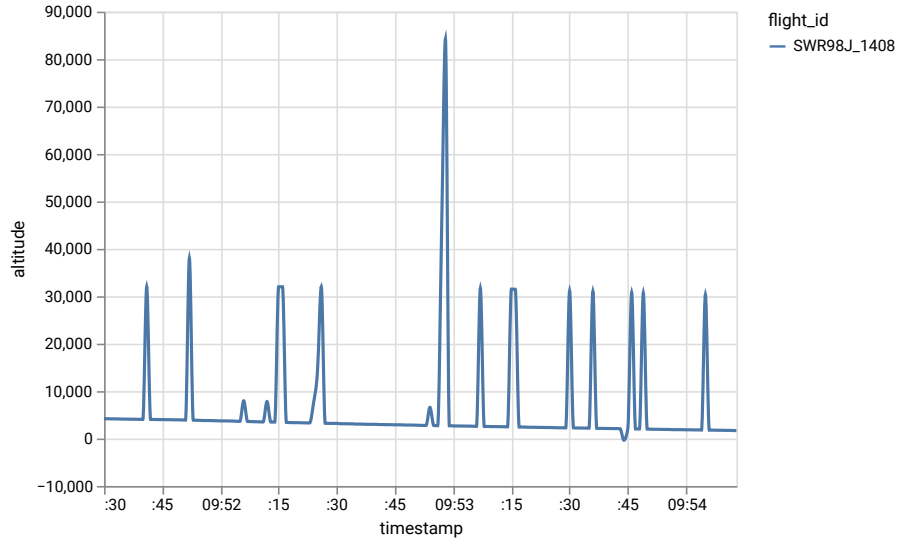


Figure 4: An example of erroneous altitude values on a flight trajectory when landing. The spikes in altitude are due to there being errors in the altitude information. This is an extraordinary sample as there are not as many glitches on all flights. The altitude is in feet.

2.3.2 External Data

As ADS-B does not provide all information necessary for a full environmental representation, more external data types are required.

Extended Mode S is similar to ADS-B; it also broadcasts aircraft messages. However, since fewer aircraft are equipped with Extended Mode S transmitters, it is harder to get a hold of as much Extended Mode S data as ADS-B. Furthermore, the amount of data an aircraft send through Extended Mode S varies. Extended Mode S can send information such as indicated airspeed, true airspeed, Mach, altitude and weather information. The major limitation to receiving Extended Mode S data is that there is no checksum in place. The lack of error detection can result in undetected transmission errors and thus erroneous values that are recorded [11].

As for runway location and elevation, they can be received from databases widely available on the Internet. Using public databases to determine runway locations is problematic as there is a lack of validation possibilities on the data. The validation may be necessary for observing whether the positional data is correct. Airport and runway location are required to determine how an aircraft positions itself relative to the runway.

For receiving weather information at the airport, a standardised and widely used format is called METeorological Aerodrome Report (METAR). This format can be decoded and applied for receiving a current understanding of the weather at an airport. METAR data can indicate the number of clouds and their height, visibility, wind speed and direction, barometric pressure, dew, and temperature. METAR messages may also indicate special conditions such as lightning [12].

The update frequency of METAR data is once an hour or half-hour. This refresh rate can although vary. If there is a sudden change in weather, the airport will generate new METAR data from the current weather standing outside the scheduled measurements. The data collected to produce METAR messages are received from weather sensors at the airport or by observers [12].

2.3.3 OpenSky Network

The OpenSky Network facilitates a database of historical ADS-B data. The database is queried through the Cloudera Impala engine [13]. For research purposes, access to the database is free. The database contains ADS-B data back to the year 2017, which covers large parts of the world. Therefore, there exist a large amount of available flight data as ADS-B transmitters sends a message every second [6].

The OpenSky Networks adds aircraft data to the database when sensors pick up the information. These sensors are placed throughout the world and receive the ADS-B information from the aircraft. The sensors are hosted by volunteers, industrial supporters, and academic/governmental organisations which sends them through the internet to the OpenSky networks database. The OpenSky Network is a non-profit organisation located in Switzerland [14]. There are regions where no information from aircraft ADS-B OUT can be received and thus are not available in the database.

2.3.4 Traffic Library

A library in Python called Traffic has an interface to the OpenSky Networks historical database. The library allows for processing air traffic data. Furthermore, it allows simple querying of the database [15]. The library can query additional information such as airport locations and their runways from another database. The runway information from Traffic includes the runway bearing and identification (e.g. 09R, 26L). The Traffic library can also append the type code of an aircraft to the data. Having a type code allows for filtering of a specific aircraft, e.g. Airbus A320 or Boeing 747. The data in the Traffic library consist of 3 types. One type is a "flight", which is a flight trajectory consisting of a series of ADS-B measurements. The second type is a "Traffic" type, which is a bundle of flights. Lastly, there is an "Airspace" type [16].

2.4 Normal Distribution

A distribution represents all the possible values a stochastic variable can take and the probability it will have a particular value. Functions can represent distributions, Where one of the most common is the normal distribution [17].

The normal distribution is also known as the Gaussian distribution. The distribution can be constructed by calculating the mean and variance of a set of values. It is used to model a distribution of continuous variables. The function for normal distribution can be seen in Equation 1, [18].

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \tag{1}$$

Where μ and σ is the mean and standard deviation respectively. The standard deviation is the square root of the variance [18].

The mean is expectation value. The Standard deviation represents the spread of the distribution. The function 1, is a probability density function (PDF) for normal distribution. It is used for determining the probability of a random variable appearing in an interval [17].

2.5 Polynomial Curve Fitting

Polynomial curve fitting generates a continuous function that represents samples of real data. The goal of this function is to create a good generalisation for predicting new data. This function is made of linear and polynomial expressions. The function is fit by polynomial regression, where the order of the polynomial expression changes the fit to the real data. Increasing the order generally increases the fit to the real data. Although, at too high orders, over-fitting can occur, which may cause the function not to generalise to new measurements [19].

2.6 Simulated Environments

Simulated environments are representations of the real world in a controllable environment. These environments allow for testing system functionality and manipulation of variables to thoroughly analyse their effect on the system. In the aerospace industry, simulators train pilots and act as validation evidence when developing new systems [20]. Further, it allows for testing different flight scenarios for the developed system. For these applications, the applied simulator for this thesis is X-Plane [21]. As mentioned by the creators about the simulator, "X-Plane is not a game, but an engineering tool that can be used to predict the flying qualities of fixed- and rotary-wing aircraft with incredible accuracy" [21]. X-Plane have been used in domains such as world-leading defence contractors, air forces, aircraft manufacturers, and space agencies.

2.6.1 X-Plane Specifications and Limitations

The X-Plane simulator allows for realistic simulation for aeroplanes in their operating environment. Features provided by the simulator are: [22]

- Scenery, which cover the earth from 74 north to 60 south latitude
- 35000 airports
- Controllable weather
- Prediction of aircraft handling
- Valid source to apply real aircraft data to
- Support for plugins

These features provide an environment that can be manipulated to create a balanced distribution for the dataset. However, there are multiple limitations when using the X-Plane for generating datasets. These limitations concern the applicability of the system to real aircraft and environments. The simulated environment lacks details present in the real world. Details of the objects, earth map and weather; differs in both textures and how they affect the environment. This difference might affect the accuracy of the neural network to detect characteristics in real scenarios. The simulator is also limited in what it can simulate; it cannot represent all possible scenarios that can occur in real life [22].

2.6.2 Simulated Scenic Scenarios

Scenic [23] is a domain-specific probabilistic programming language that creates scenes in simulated environments. The language allows staging of the scene based on the simulator's capabilities and coding to generate datasets used for training CNN. Simulators currently supported by Scenic are Carla, X-Plane, Grand Theft Auto V, LGSVL, and Webots [21], [24]–[27]. In these simulators, Scenic programming allows for coding specific conditions that need to be fulfilled in the simulation. For example, in X-Plane Scenic programming can specify simulator weather and position of the aircraft on the runway to create that scene. The programming can also add randomisation to the scenarios, which, if bounded correctly, represent a valid dataset source when training NN. The workflow when defining scenes utilises configuration files, Scenic file and a sampler. Configuration files are YAML files that state parameters that need to be known for the generation process. These parameters include latitude and longitude coordinates for the runway as well as the aircraft's position. The Scenic file defines boundaries for how the scene shall be sampled when generating datasets. These boundaries are set based on normal, uniform, random and discrete distributions. For example, weather defined as Uniform(0,1,2,3,4,5) will result in the weather condition 1-5 equally distributed throughout the samples. The current iteration of Scenic with X-Plane only supports simulations on runways. Thus, the generation system needs further development to support landing scenarios.

2.6.3 X-Plane Connect

To represent the scene in X-Plane given by the Scenic generation process, X-Plane Connect (XPC) [28] is used. XPC is an open-source X-Plane plugin developed by NASA that allows for control over the X-Plane simulator. This control is established by altering datarefs which define the current state of the simulators environment. Datarefs exist for most functionalities in the plane and the environment around it. The full list of datarefs available can be seen in [29]. XPC also provides predefined functions which change multiple datarefs at once [30]. For example, sendPOSI sets the aircraft's position and orientation according to latitude, longitude, altitude, pitch, yaw, and roll.

2.6.4 X-Plane Coordinate Systems

X-plane uses two different coordinate systems to position the plane in the simulator. The first is global coordinates which are represented by latitude, longitude and elevation. Latitude position is the rotation around the centre of the earth towards the poles. 90+ degrees represent the north pole, -90 degrees represent the south pole, and 0 degrees is the equator. The longitude position is the rotation around the centre of the earth parallel with the equator where 0 degrees is located at the royal observatory, Greenwich, England. These coordinates, in conjunction with an altitude, can specify any position on the earth's surface. In X-plane, this coordinate system is not very precise, and thus, OpenGL coordinates are used. The OpenGL coordinate system is a cartesian system that positions the plane shifted in any direction according to a reference point in metres. X represent west-east, where the positive X-axis is aligned east. Y represents downwards-upwards, where the positive Y-axis point upwards. Z represents south-north, where the positive Z-axis is aligned north. The simulator assigns the reference point to a position reasonably close to the aircraft. This coordinate system is more precise than the global coordinates due to its ability to move the aircraft a set distance in any direction. However, minor errors occur with the linear position change due to the curvature of the earth [31].

2.7 Artificial Neural Network

Artificial Neural Network (ANN)s, generally referred to as Neural Network (NN)s, are computing systems inspired by biological neural networks. Biological neural networks are complex and not entirely understood by scientists today. These networks operate by storing memory and neural functions in neuron and their connections. New connections between neurons or alterations of existing connections are achieved through learning. When considering computer-based NN, they learn through applying mathematical optimisation algorithms to find patterns and similarities in inputs. A simple model of a neuron is visualised in Figure 5. This model has a scalar input (in) which is multiplied by the weight(w) of the neuron. Adding this with the bias(b)(if there is one) result in a net input(ni). The net input is then inserted in a transfer function that aims to satisfy the problem which the neuron is attempting to solve. This transfer function shows the effect of input, weight and bias, which is used to find similarities between them. The neuron can use different transfer functions to analyse different specifications of the problem [32]. The transfer function also includes an activation function that defines whether a neuron shall fire or not. This function determines if the calculation is reasonable to be used in the network [32]–[34]. A NN consists of multiple neurons in three kinds of layers; input layer, hidden layers and output layer. If multiple hidden layers exist in the network, it is considered a Deep Neural Network (DNN). Each layer consists of numerous neurons which are connected to all previous and later neurons. Chaining multiple layers allows for more complex decision making in the NN. Training these networks is conducted through alterations of weight and biases in each neuron based on datasets [34], [35]. These alterations are performed through backpropagation, where calculations of errors allow identifying the adjustments needed [33], [34]. When using NN for vision-based object detection systems, Convolutional Neural Network (CNN) are commonly used.

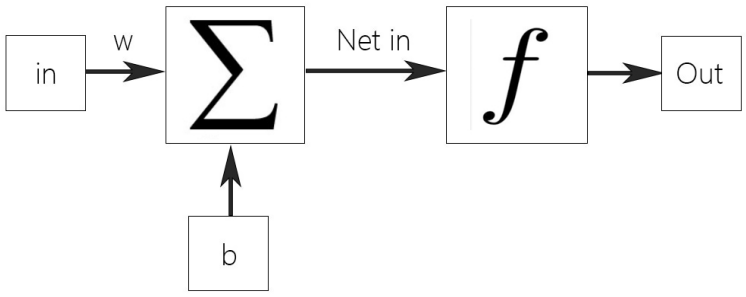


Figure 5: This figure visualises the functionality of a single neuron in a neural network. The neuron receives an input (in) which is multiplied by its weight (w). A bias (b) is then added to create a net input. The net input is inserted in a transfer function that identifies characteristics based on the net input, resulting in an output(Out).

2.7.1 Convolutional Neural Networks

CNNs are vision-based neural networks that contribute to the state of the art computer vision domain. These networks provide automatic feature extraction through the functionality of object detection, segmentation, and image classification, which improves self-awareness of the system [36]. CNNs can have different architectures, but their essential components are similar. For example, the CNN LeNet-5 [37] contains the basic layers convolutional, pooling and fully connected layers. The convolutional layers learn feature representation of the input image [38]. A kernel element utilises pixel values and matrix multiplication to extract these features, thus resulting in a smaller matrix resembling the image's features. The pooling focus on the reduction of the spatial size after the convolution layer. This layer reduces the computational power needed to extract features and reduces the dimensionality [39]. The convolutional and pooling layers are performed multiple times until one or more fully connected layers of neurons are used for the reasoning of the matrices [38], [40].

2.7.2 Object Detection with Neural Networks

Object detection aims to identify specific features of interest in the images. Using DNNs for object detection has shown outstanding performance recently. This performance is due to its ability to learn complex models to create powerful object representations. However, for this approach, considerations need to be taken for classifying features and their position on the image [41]. The workflow for traditional object detection follows three stages; informative region selection, feature extraction and classification. Informative region selection focuses on analysing what parts of the image are of interest when finding specific objects. If such a region can be identified, object detection only needs to consider this region to find the feature. This method reduces the computational power need for object detection. Feature extraction aims to provide a robust representation of the object of interest. This representation considers features of the object, which defines and separates it from others. Using these features allows for improved detection capabilities for different objects. Classifications focus on differentiating objects to different categories. This differentiation allows for the clustering of similar objects, and a clear definition of classes present in the environment [42].

2.7.3 Dataset Coverage

The performance of a NN for vision-based object detection is heavily affected by the quantity and quality of the dataset. The dataset needs to represent the NN operating environment to its best extent. The NN can then learn the desired features of the environment. Using a balanced dataset covering all possible scenarios with great detail over the environment would achieve this. However, creating such a dataset is a complex task and requires a massive amount of data and processing

power [43]. This task is a common problem for dataset industry. Class imbalance is a real world property; thus, it results in an imbalance when gathering datasets from it [36]. For example, when considering animals, there is far more dataset for common compared to rare animals. Thus, it results in a higher possibility of creating a balanced dataset for detection of common compared to rare animals [44]. This problem also translates to landing scenarios in the form of specific landing manoeuvres. As mentioned in the landing approaches section 2.1, landing manoeuvres are heavily affected by the environment, which needs to be considered in the dataset coverage.

2.7.4 Dataset Balancing Methods

To limit the amount of data but still have great coverage over essential characteristics, sampling methods can be applied. These methods are divided into two categories, data-level and algorithmic-level. Algorithmic-level utilises Machine learning (ML) to accumulate imbalanced data. Data-level focus on manipulating the data through oversampling and undersampling [43].

The oversampling approach focuses on duplicating samples of less common scenarios in the dataset. These scenarios are then more likely to be interpreted by the neural network. However, this could result in overfitting where the dataset is too specific, limiting the neural network's performance [45]. Different approaches which utilise oversampling have been researched to improve its balancing features further [36]. The dataset can be divided into different clusters which represent each class of interest in the environment. By oversampling each cluster separately, the dataset will, with more detail, describe the features of the classes [46]. Oversampling can also be combined with boosting algorithms to find and fix flaws in the dataset. Boosting algorithms can find hard examples from the dataset and utilise oversampling to generate synthetic data to balance it [47]. The undersampling approach focuses on removing common samples to balance the dataset. This method can be problematic since it can remove useful dataset samples, which limit detection of specific cases [45]. To mitigate the loss of useful data, one-sided selection can identify redundant data that is too similar and remove it [48].

2.8 Detectron2

Detectron2 is a library or platform that enables access to object detection and segmentation algorithms. It is developed by Facebook AI Research (FAIR) and is open-source [49].

The Detectron2 platform is built upon Pytorch [50]. Pytorch is a Python library that enables computation with GPU acceleration and building neural networks [51]. Models such as Faster R-CNN, Mask R-CNN, RetinaNet, and DensePose are available in Detectron2 [50].

2.9 Evaluating Object Detection

To evaluate an object detection model, a validation/test dataset is necessary. There needs to exist ground truth information for this dataset that indicate the true position and classification of the objects in the images [52].

Evaluating the performance of object detection models commonly revolves around one metric, the Mean Average Precision (mAP) or if there is only one class in the dataset, it is known as Average Precision (AP) [53].

The AP score is determined by the aid of three values; precision, recall and Intersection over Union (IoU). The precision is the ratio between the number of true predictions and all the predictions, as seen in Equation 2. The recall is instead the ratio between the true predictions and all that is true, whether it be predicted or not. The formula for recall is described by Equation 3, [53].

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{2}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{3}$$

The third necessary value for determining AP is IoU. IoU is also a ratio. IoU is calculated by determining the overlap and union between two regions, the true region and the predicted [54]. If

the predicted region fully contains the true, it would have an IoU of 1, i.e. a full overlap. The formula for IoU is 4.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{4}$$

With the aid of the three values, the AP can then be determined. The IoU value is used for determining if a prediction is a true positive, false positive or false negative. The threshold for the IoU has to be set for producing a true or false value. An IoU above the threshold results in a true positive. A value below the threshold is a false positive, and a false negative is when there is no detection. Once these are established, the precision and recall can be calculated [53].

Precision and recall can then be plotted against each other and then smoothed. Smoothing is necessary for mitigating the effects of smaller variations. The AP is determined by calculating the area under the precision-recall curve as seen in Equation 5, where p is precision and r is recall [54].

$$AP = \int_0^1 p(r)dr \tag{5}$$

2.9.1 Detection Evaluation with COCO

Microsoft Common Objects in COntext (MS-COCO), or just COCO, is a large dataset for use in object detection and segmentation [55]. However, it also carries with it annotation fileformats and detection evaluation metrics that Detectron2 uses.

COCO, evaluation includes basic AP measurements. However, it also includes an AP metric which is different. This metric consists of calculating AP at several different thresholds of IoU, then averaging all of them. The IoU threshold starts at 0.5 then ends at 0.95 with a step size of 0.05. This metric, according to [56], is the most important, at least when it comes to performance on COCO. This metric is denoted as the primary challenge metrics in COCO [56].

Metrics that COCO evaluation in Detectron2 reports when an object detection model is run on a validation set:

- AP IoU from 0.5 to 0.95
- AP IoU at 0.5
- AP IoU at 0.75
- AP Small Region
- AP Medium Region
- AP Large Region

2.10 Collaboration

This master thesis was performed by students from the Dependable Systems program in collaboration with SAAB Aeronautics, Järfälla, Stockholm. SAAB strives to keep people and society safe through their technological solution from military defence in all domains to civil security. Saab operates in all continents with continuous adaptation to match the continuously changing world. Due to the advances in neural networks used for object detection, its application to the aviation industry has become an interest of the company. The motivations for this thesis was to analyse the effect of using synthetic generated datasets for detecting runways using object detection. This would, in turn, provide a knowledge base for SAAB that can be utilised when developing object detection systems.

3. Related Work

This section presents other research works that concern methods and results related to this thesis. In the following sections, papers are introduced regarding receiving and filtering flight data, creating and applying synthetic data, and ways of detecting and tracking runways.

3.1 Aircraft Information

Several papers research the landing approaches with the aid of ADS-B data or equivalent technology. Aircraft data can be received from Correlated Position reports for a Flight (CPF), which is data derived from ATC surveillance systems as shown by Peeters and Guastella [9]. The difference with ADS-B compared to CPF data is that ADS-B has a higher update rate; however, the data from ADS-B is less stable since the data contains more glitches [9].

The OpenSky network receives ADS-B information. Furthermore, the OpenSky networks facilitate an Application Programming Interface (API) that allows researchers free access to a database with ADS-B data, as demonstrated in [10], [57]–[59]. Lemetti et al. [10], [59], describe applying the OpenSky network for discovering and examining the possible causes for delays and inefficiencies when departing or arriving at airports. Lemetti et al also use two different databases for comparison. These databases are the Demand Data Repository (DDR) hosted by Eurocontrol and OpenSky's historical database. The outcome of this comparison is that the OpenSky network has 800-1000 points inside the Terminal Maneuvering Area (TMA), while the DDR has 10-15 points.

There are issues with using the OpenSky network, or ADS-B data as seen in [9], [10]. The data quality of ADS-B samples are not stable; there are glitches in both vertical and horizontal data from ADS-B [9]. Lemetti et al. [10], describe issues with ADS-B as repeated waypoints, infeasible time stamps and the positional coordinates being off from the trajectory. As mentioned in [9], [10], merging ADS-B, and other similar kinds of data ensures more stable and detailed trajectories.

As demonstrated in [10], [58], [59], the defined TMA is used to set the bounds in where the aeroplanes arrive and depart from the airport. The data sampling occurs when the plane is inside the TMA to ensure the data only contains ascending or descending trajectories. There is no description of what querying conditions were given to the OpenSky database to receive flight ADS-B data from the TMA.

There are drawbacks with using the OpenSky network as a source of flight data. Lemetti et al. [59], mention that there is a lack of aircraft types and identifier data. However, Lemetti et al. still recommend the OpenSky Network as it is a valuable data source for research with a high accuracy demand.

Kamsing et al. [60] apply ADS-B data and describe the goal is to determine the landing routes at Suvarnabhumi International Airport in Thailand by the aid of ADS-B data. Similar to [10], [58], [59], flight ADS-B data comes from the OpenSky network when the aircraft is in the proximity of the airport. Kamsing et al. [60], describe the conditions set on the data to only receive trajectories that contain landing or departing at the airport. Some of those conditions bound an area that envelops the airport, filtering based on origin and time. Kamsing et al. [60], describe using K-Means clustering for defining the routes.

3.2 Synthesized Data

A report published by EASA [1] describes that for design and testing of machine learning models, simulated or synthesised data is likely to be needed when applying them for safety-critical implementations. There are three example categories of synthesised data described by EASA [1]. These are basic transformations of real data, more advanced transformations of real data, and fully or mostly synthetic data.

Basic transformations of real data consist of data augmentation. This augmentation includes applying translation, rotation, flipping, and more. It also includes changing brightness, noise and hue.

Examples of more advanced transformations of real data include pasting objects into images, such as adding aircraft or cars. This method requires no need for manual annotations as the place of the object locations are known.

The last category is fully or mostly synthetic data. This category is the highest in complexity or syntheticity. An example from the report is a 3D urban scene with components that come from the real world to some extent.

The report from EASA [1] also states that it is essential to have sample data from the target space for creating models which perform well for the operational phase. This revolves around the problem of applying synthesised data to the real target.

Levin and Vidimlic [61] apply two methods for synthesised data described by EASA [1]. The data used for synthesis were of real images during a landing approach. For generating these synthesised images, both augmentations and the cut-and-paste method are applied. Levin and Vidimlic also applied augmentation in the form of luminance, motion, weather and artefacts.

For weather and lighting, a Python library called Imgaug is used to apply effects to the images [61]. This library allows for putting an overlay over images with various effects. Conditions used for weather were fog, clouds, rain and snow. Adding weather and lighting yielded an 18.1% reduction in AP score in Levin's and Vidimlic's work. The AP score used for this was COCO.

3.3 Runway Detection and Tracking

Several authors describe various methods to detect and track a runway, including edge detection, colour segmentation, and deep learning algorithms.

Ajith et al. [62], use camera images to identify and track a runway. The intent is to detect the runways from an aircraft or Unmanned Aerial Vehicle (UAV) landing approach. A UAV will have a different landing approach compared to an aircraft since some UAVs can take-off and land vertically. As for the procedure for detecting and tracking a runway, the method involves using colour segmentation.

The first stage is to identify the runway, which is implemented by looking at the surrounding, runway colours and runway characteristics. Compared to the surrounding, the colour of the runway is the most distinguishable feature that allows separation of the runway to the environment. Once the runway is detected, the tracking stage can begin. The tracking continuously updates a template of the runway received from the detection stage to ensure constant runway boundaries for indication [62].

Ajith et al. [62] describe using several videos in testing the algorithm. Two videos were from real landings, and the authors also describe testing the algorithm on videos created in Google Earth, which simulated landings.

Wang, Li and Geng [63], apply a Canny edge detection algorithm instead of colour segmentation to identify and track runways. The focus of the paper is to limit the landing approaches for UAVs. For a UAV runway, the boundaries are always straight lines no matter how much translation, rotation or zooming is applied. Wang, Li and Geng [63], conclude that the method is suitable for UAV automatic landing by visual navigation [63]. The authors describe constructing a simulation to test the algorithm. The simulation replicated a scenario where a UAV is landing [63].

Deep learning is applied for runway detection, as demonstrated in [64]. Akbar et al. [64] focus on using runway detection and tracking for UAV usage with fixed wings. A fixed-wing UAV requires a runway. Similar to [62], [63], the imagery comes from a camera.

Akbar et al. [64], detail that most research on runway detection methods previously are not based on machine learning. This lack of machine learning implementation is due to the restrictions on real-time object detection methods. Although, due to the recent increase in computational power, it may be possible for real-time applications [64].

Similar to what the authors have described previously, the procedure branches into two stages. Firstly the detection stage to find whether there is a runway present in the image, and lastly, localisation to determine where the runway is in the image. The detection stage applies only a CNN, and the localisation stage implements both a CNN and a line detection algorithm. The dataset used to train the CNN consists of satellite imagery from Google Earth [64].

4. Method

This section describes all the processes and measures taken to answer the research question. All research questions will be answered by experimentation.

4.0.1 RQ1

Research question 1 is as follows, "**What method of generating distributions from real data produces the highest prediction accuracy from generated images?**".

The methods that generate these distributions from real data are one move, poly, cube normal and true. These methods are described in Section 4.3. The real data is the flight data coming from ADS-B during the landing approach. The prediction accuracy refers to the Average Precision (AP) score from evaluating an object detection model on a set of images. These images are referred to as "generated images" in the research question; these images are generated in the flight simulator X-Plane.

This research question is dependant on five parts. These parts are visible in Figure 6. First, the flight data is received for just landing approach. This data is then given as input to 4 methods that produce new flight data distributions, aiming to replicate the real flight data. The Scenic language then creates new flight data from these distributions made by the four methods. By this new flight data, an image can be generated in the X-Plane simulator by placing an aircraft at the positions produced by Scenic. These images are then annotated for producing information about the location of the runway. This step of generating images based on distributions occurs for each method. Once four sets of images are created and annotated, they are used for training neural networks that produce object detection models. There are four object detection models, one for each method of creating distributions from real flight data (One move, true, Poly and Cube Normal). The last step is for evaluating these object detection models. A new set of images is produced based on separate flight data from those used previously during the training. These images are produced by the "true" method, which is simply the real flight data 1-1 with the new flight data. This new set is displayed in the Model 6 as "Flight Data Airport 1 New Set". The four object detection models can then be evaluated against this new set of images by comparing the true position of the runway from the annotation and those predicted by the models. The evaluation generates AP scores. The object detection models with the highest AP-score is then the answer to the research question. Figure 6 provides a visual block diagram of this procedure.

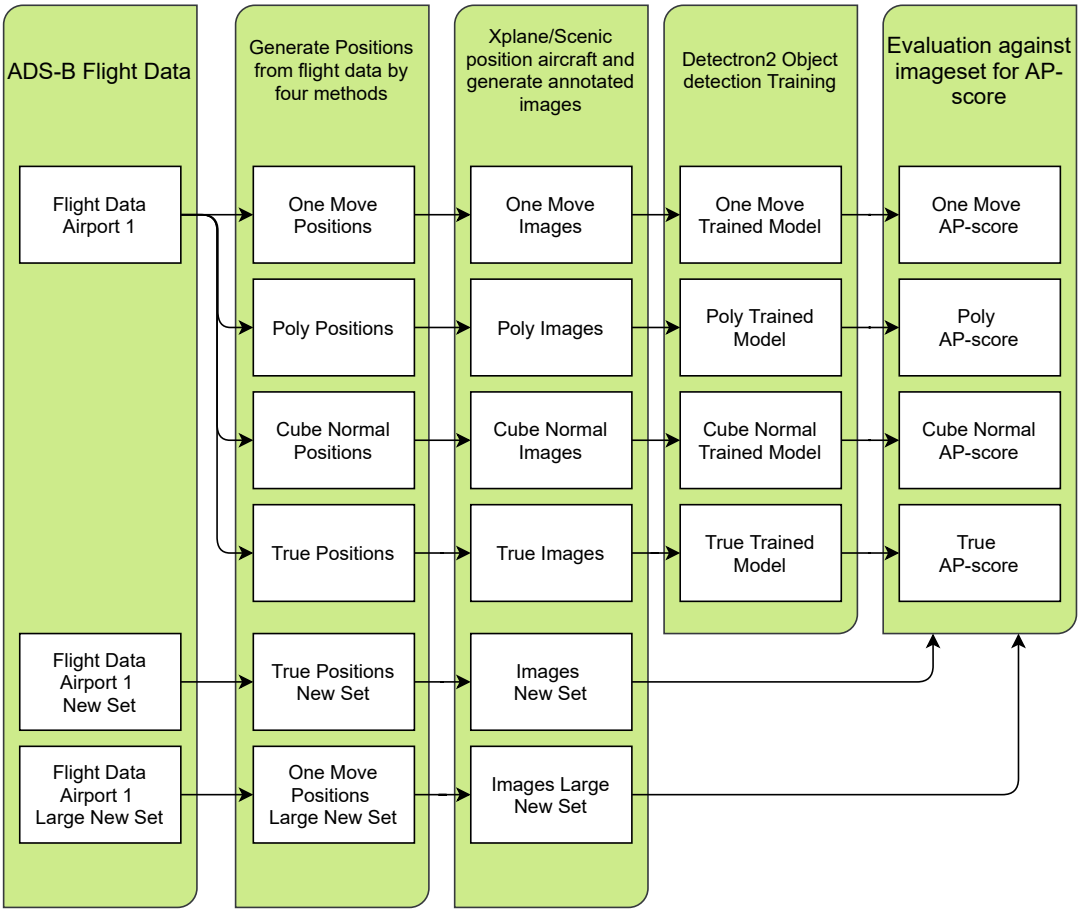


Figure 6: Model displaying the process for answering research question 1.

4.0.2 RQ2

Research question 2 is as follows, "**How does changing airport runways affect the prediction accuracy of the object detection?**"

Similar to research question 1, the prediction accuracy also refers to the AP-score derived from evaluating an object detection model with an image set. A visual representation of the steps taken to answer this question is seen in Figure 7. Here three types of flight data are sampled from an ADS-B database. The difference between these three sets of flight data is the airport's location where the landing approaches took place. For each airport, the generated flight data places the aircraft in a simulator. When the aircraft has been positioned, the image for that view can be taken. Only the true positions of the flight data are used, no methods replicating distributions, which was the case for RQ1. Once image sets for each airport have been created and annotated, they are used for validation against two trained object detection models. This validation against a model generates the AP-score. There are two models used for this evaluation; both run on all the images produced. The object detection models are trained on only one of the airports and they are separate from those in RQ1. The validation images from the airports are then used during the evaluation with the models for generating AP scores. The difference of the generated AP (prediction accuracy) between the airport which was trained on and those without then determines the effect, i.e. answer to the research question.

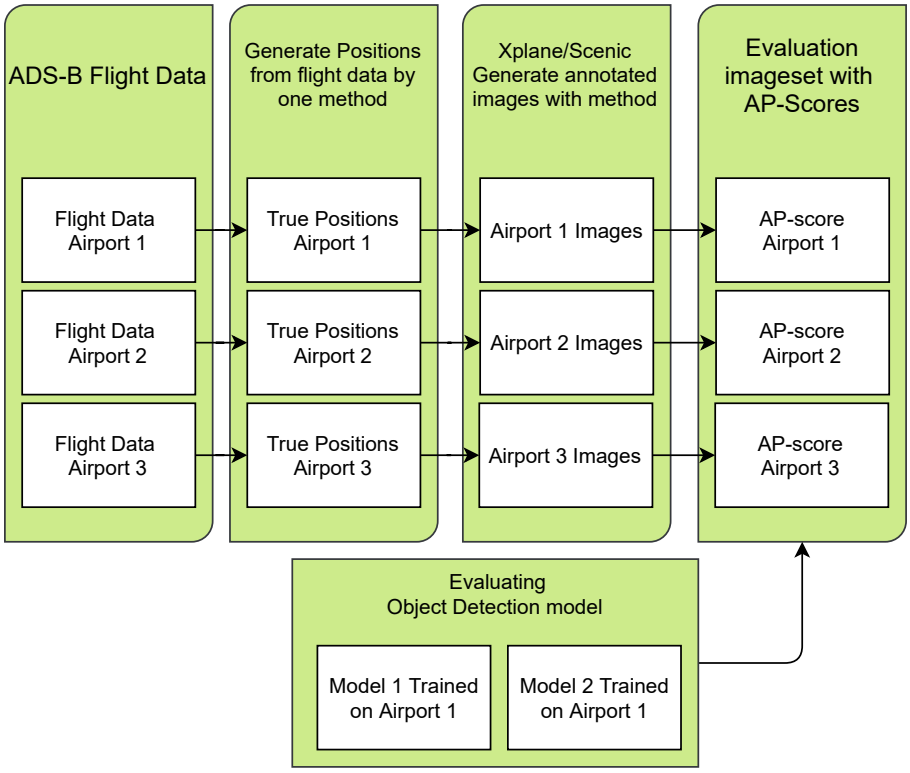


Figure 7: Model displaying the process for answering research question 2.

4.0.3 RQ3

Research question 3 is as follows, "**How does environmental conditions affect prediction accuracy of the object detection?**".

The environmental conditions in this research question consist of 3 types of conditions. These are changes in clouds, time and rain. All the conditions are varied by manipulation in the simulator, i.e. the flight data and methods for generating distributions are all the same. Hence, the position of the aircraft is the same for all the images in the simulator. Image sets are produced for each environmental condition and one additional for comparing with no environmental condition. This image set with no change in environment is necessary for determining the impact of the other image sets with conditions. Similarly to RQ1 and RQ2, AP-score is used for establishing the effect, i.e. the prediction accuracy. As with RQ2, two object detection models are also used. For RQ3, the models are instead trained on images with no environmental conditions. Therefore, applying environmental conditions can then determine the effect they have on untrained models. The AP-score is then generated by evaluating these models on images with and without environmental conditions present. Then by comparing the AP-score of the images with no environmental condition and those with, the research question can be answered. The methods for answering the RQ is shown in Figure 8.

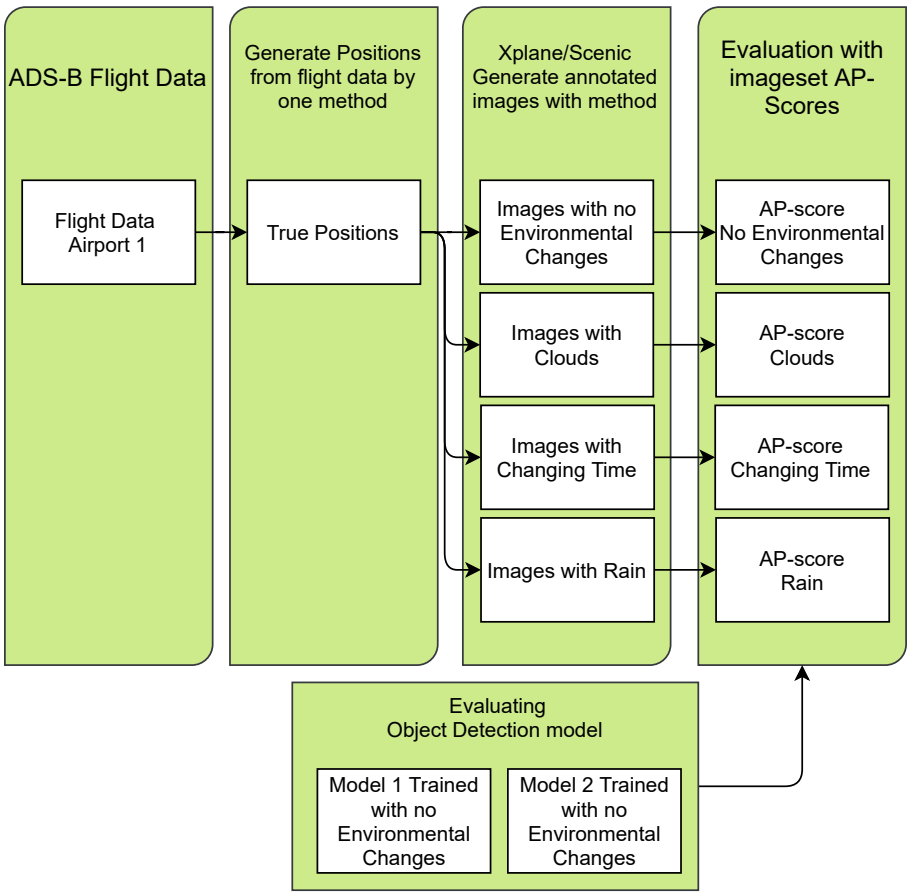


Figure 8: Model displaying the process for answering research question 3.

4.0.4 RQ4

Research question 4 is as follows, "**How does aircraft attitude affect prediction accuracy of the object detection?**".

This question is the same as RQ3; however, instead of changes to environmental conditions, it is attitude changes. Attitude is made up of three components, yaw, pitch and roll; it also has an image set for a baseline, with no attitude changes. No changes to the attitude are defined as zero pitch, zero roll and heading direction parallel with the runway. The difference between the generated AP-score from the attitude image sets compared to the ones with no changes, answers the RQ. A model for this experiment is visible in 9. The object detection models are trained on images with no attitude changes present.

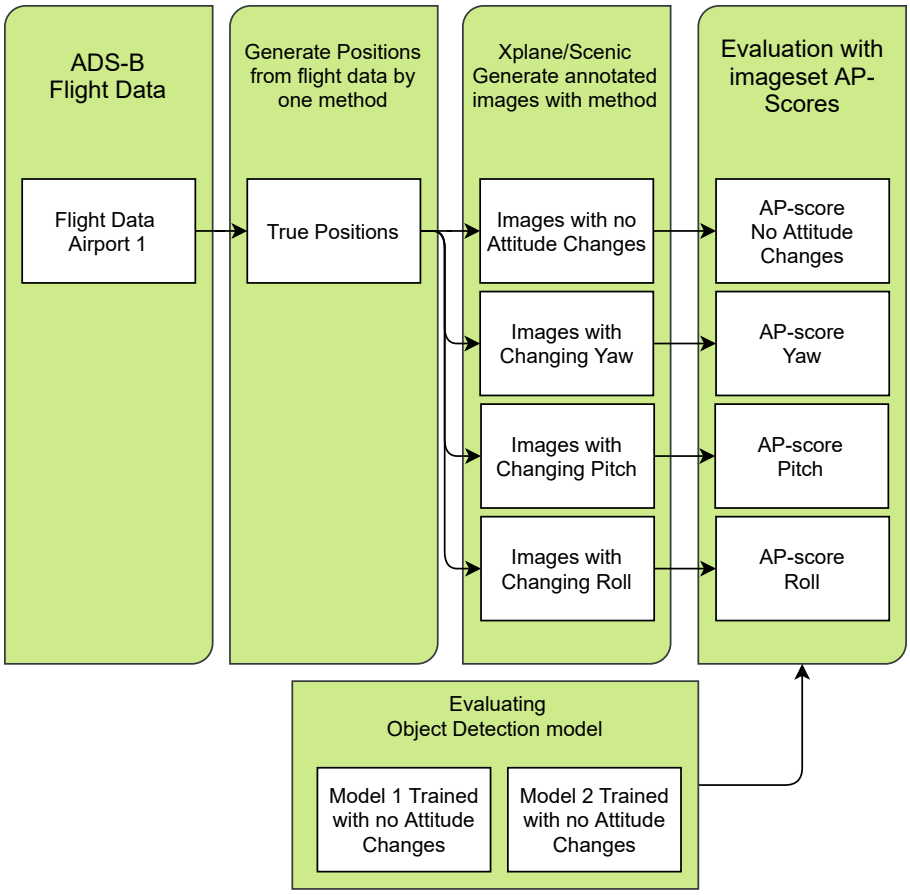


Figure 9: Model displaying the process for answering research question 4.

4.0.5 RQ5

Research question 5 is as follows, "**What variables need to be considered when generating reliable datasets used for safety-critical landing approaches?**".

Answering this research question is dependant on the results of questions 1, 2, 3, and 4. The previous RQs focus on analysing how different variables affect the detection capabilities of the neural network. This RQ will analyse how these variables need to be considered and applied for creating a reliable dataset used for safety-critical landing scenarios. The variables analysed are limited to those considered by previous RQs.

4.1 Flight Data

As answering the research questions is dependant on flight data, it is necessary to receive and analyse real flight data during landing approach. The chosen data originating from the flight system ADS-B. ADS-B is used for the flight data as it contains a significant amount of data, and it is readily available from accessible databases online.

4.1.1 Requesting Flight Data

The library Traffic allowed for downloading the flight data. The library has an interface to the Opensky Network historical database which contain the ADS-B flight data. In the query made possible by Traffic, conditions were set on the flight data downloaded. These conditions were both the start and endpoint of the flight data in time and the flights' arrival airport. Where the arrival airport is set given the ICAO-code of the airport. The flight data time interval is necessary since it limits the amount of data to be downloaded. The time to query the data is dependant on the size of the data. Therefore, wider timeframes in the query take a longer time.

Querying for the arrival airport ensures that it is only landings at the airport, removing most of the take-off data, excluding the flights which take-off and land at the same airport. Furthermore, querying for arrival airport ensures there are only flight trajectories that are connecting to the airport.

The downloaded flight data contains ADS-B data which have data spread over wast regions of the world. The data includes different flights from different departure airports to the airport queried for landing.

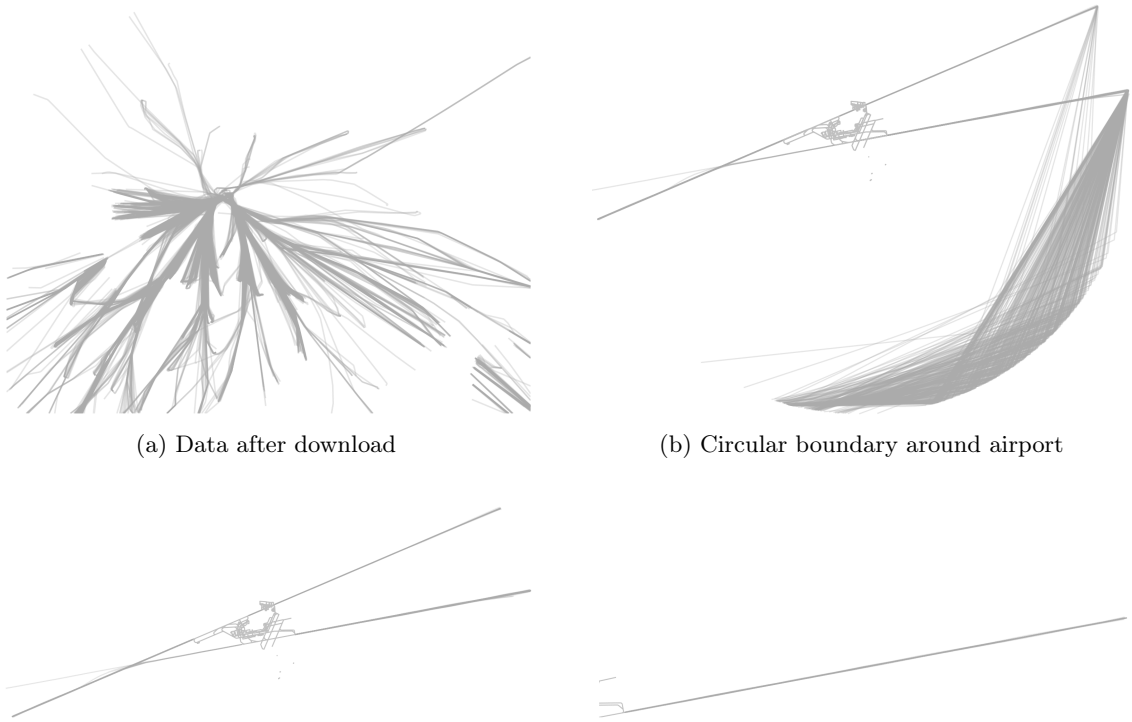
Extended Mode S data could not be downloaded from the OpenSky Network as there were issues with availability. This data, together with ADS-B data, could have enriched the data to contain more flight information.

Downloading flight data from march 2020 and onward requires wider timeframes. As there are fewer flight data due to the Covid-19 pandemic impact on air traffic [65].

An issue seen with the downloaded data for some airports is the lack of ADS-B data when an aircraft is at a low altitude around the airport. This lack of flight data is due to there not being an ADS-B receiver in the airport's proximity connected to the OpenSky Network. To counteract this, only airports that have OpenSky connected receivers are chosen. By looking at the OpenSky Network Explorer and sorting by aircraft on the ground, only airports with receivers can be discovered. An airport with no receiver would not pick up the ADS-B signal due to the aircraft being too low. An issue with this method is not knowing when the receiver was installed. Querying flight data before the receiver was installed results in no data at a low altitude around the airport. A solution is querying data closer to the present time, although this causes issues as there is a lack of flight information due to the Covid-19 pandemic [65]. The interval of time is set dependant on how busy the airport is. The range used for the time was between 12 days to 2 months. The timeframes used during the query of data is mostly located in the year 2021. This time is used because the amount of flights has somewhat increased; even though there are still restrictions from the Covid-19 pandemic, they are not as significant as those located in 2020. Furthermore, the probability of there not being a receiver installed at a chosen airport is minimised when closer to present time.

4.1.2 Filtering for Landing Approach

After requesting data, the result contains unnecessary flight data that is not relevant to the landing approach. Since after the query, the data include the entire flight trajectory from start to end, as shown in Figure 10a. Only the end of the flight trajectories is necessary, as the landing approach is at the end of the flight. The Traffic library allows for filtering the data. Many steps made during the filtering of flight data were motivated by the methods demonstrated in [66].



(a) Data after download

(b) Circular boundary around airport

(c) ILS filtering

(d) Final data after filtering

Figure 10: Steps taken to filter the flight data for only landing approach. The cluster in the middle of the images in 10b and 10c is Paris-Orly airport, LFPO. The airport in Figure 10a is where the lines converge into. In Figure 10d the airport can be seen in the bottom left. The lines in the figures are flight trajectories, which comes from the flight data. Figure 10a, covers trajectories over wast regions of Europe, hence it is zoomed out significantly.

A circular boundary with the centre at the airport allows filtering for the flight trajectories' ends. All the flight data outside this circle is removed. To determine if a flight is inside the boundary, the distance between the flight's coordinates in longitude and latitude and airport's coordinates is below 9 nautical miles. The circular boundary radius is set at 9 nautical miles as it is inside this range where most of the landing approaches are contained. Most aircraft have turned in for a landing at this range and are aligned with the runway for landing. This turnover distance from the airport to aircraft was discovered by testing and observing the flight data at different circular boundary radii. The flight trajectories after setting a circular boundary can be seen in Figure 10b.

The circular boundary cuts down on the size of data. A column is appended to the flight data that is the result of calculating the aircraft's position to the airport, which is a distance in nautical miles. Then another Traffic function queries the added column for distances below 9 NM.

An alternative to filtering for data inside a circular boundary, is filtering for data inside an airports Terminal Maneuvering Area (TMA), as demonstrated in [10], [58], [59]. The Traffic library allows this function; however, it requires access to the Demand Data Repository (DDR) from EUROCONTROL. This access could not be established as it is not given to students.

Take-off flight data may still be present, as flights that start and end at the airport are inside

the bounding circle. Therefore a vertical rate condition is set. An average vertical rate for an entire flight trajectory below -500 feet/min ensures that it is only landing aircraft data. Flights at a high altitude above the airport may still be present in the data. Therefore, an altitude condition is set. The flight data is filtered for altitudes below 10000 feet.

For further filtering only landing approaches, another Traffic function is applied, which returns flight data that are in Instrument Landing System (ILS) regions of the airport. This function only filters flight data based on a region behind the runway and if the aircraft is travelling towards the runway. Since this only filters based on a region and travelling direction, it means that a Visual Flight Rules (VFR) landing can also be present in the data. If the flight data is in this region, it is on a landing approach to the runway. This filtering takes place for all the runways at the selected airport. After filtering with this Traffic function, the flight trajectories can be seen in Figure 10c. Furthermore, the ILS function appends the runway designator to the flight data. The designator indicates which runway the flight lands on. Another Traffic function appends the ICAO type code of the aircraft to the flight data.

The aircraft type code and the runway is then selectable, allowing analysing them exclusively. However, many times only one or a few amounts of runways at an airport is used. Therefore, it is helpful to find the runway which contains the most landing flight data. A function allowed for displaying the most common runway and the most common type of aircraft at that runway. This function allows for collecting the most data from one airport.

Once runway and aircraft type is chosen, only the landing approach at one runway is in the flight data, as shown in Figure 10d.

As presented in Figure 4, the erroneous data is removed to a degree by eliminating all flight data above the altitude of the first flight in the landing approach data and below the last. This filtering removes most spikes.

All forms of resampling of the data are avoided, as resampled data changes the original data values, which may cause issues with validity. This validity issue comes from the data not being from actual data. There are functions in the Traffic library which resamples data, and they are not used. Instead, points are removed for being invalid; they are not filled in.

The issue with removing data is that some might be valid, as filtering methods may have inherent issues. This issue would cause problems with validity, as the flight data may not be encompassing. However, for this filtering, only few amounts of data are removed. Furthermore, through testing and observing the filtering methods before and after, the changes can be seen. The observation of data takes place in Google Earth or Kepler.gl, by plotting both latitude, longitude and altitude.

4.1.3 Setting Data Relative to Runway

As the coordinates from ADS-B are given in latitude and longitude, it is difficult to perform a distribution analysis, as it adds an unnecessary dimension. Only the aircraft's position relative to the runway is important, as this is used in the distribution analysis.

The plane's position relative to the runway depends on three parameters. These are lateral positioning, distance to the runway and altitude. Lateral positioning is visible in the Figure 2, where it is denoted by $d_{Craft \rightarrow Centerline}$. The aircraft's distance to the runway is also visible in the figure as $d_{Craft \rightarrow Runway}$.

The distance to the runway ($d_{Craft \rightarrow Runway}$) is appended to the flight data. It is calculated by a Traffic function that gives the euclidean distance between the flight data coordinates and a coordinate at the beginning of the runway. It only operates in 2D space; it does not take into consideration the altitude.

Figure 11: Example of where a marker is placed at the start of the runway. Note the marker is placed some distance to the left of the runway centerline indicated by the arrows. This image is taken in Google Earth.

The centerline is calculated through a coordinate on the runway, where this determines the distance and bearing. A database in Traffic allows this information. However, as the image in 11 indicates, the coordinate can be placed not directly on the centerline of the runway. This difference causes a systematic error in the data as the flight data will have a constant deviation to one side. The deviation is, however, minor, so compensating is not necessary.

The angle ($\alpha_{Craft \to Runway}$) is calculated by taking the difference between the aircraft bearing and runway centerline bearing. The aircraft bearing angle is also determined similar to the distance, instead of an angle from north. From the distance and angle between the aircraft and the runway, the centerline distance ($d_{Craft \to Centerline}$) can be calculated by the formula below.

$$d_{Craft \to Centerline} = \sin(\alpha_{Craft \to Runway}) d_{Craft \to Runway} \tag{6}$$

4.2 Distribution Analysis

The distribution analysis connects the flight data to Scenic. As Scenic generates positions based on distributions. The distributions are based on the landing approach flight data as produced in Section 4.1. There are three types of distributions necessary for replicating the flight data in a simulated environment enabled by Scenic. These three types are described in Section 4.1.3, as lateral positioning, distance to runway and altitude.

Scenic allows for generating positions based on a normal distribution. Therefore, it is necessary to analyse if the flight data fit a normal distribution since if it fits a normal distribution, it simplifies carrying flight data over to Scenic. Generating flight data by normal distribution when the true flight data is not normally distributed will produce a low fit between the two.

The distribution analysis aids in understanding how methods that apply normal distribution perform.

4.3 Methods to Replicate Flight Data

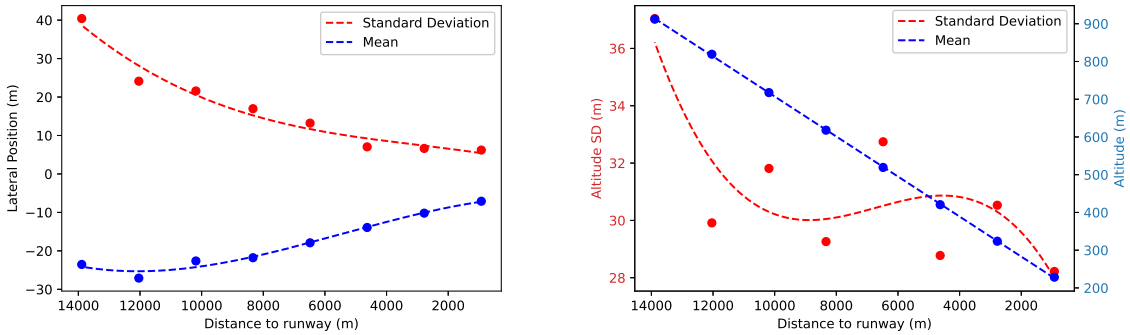
Four methods are identified as being able to replicate the real flight data distribution. The identified methods are polyfit (or just poly), cube normal, one move, and true. The names for these methods are used throughout this paper. Polyfit is named after polyfitting, i.e. polynomial regression. Cube normal is named after the flight data being sectioned into cubes, and it applies normal distribution. One move is named after it picks one point of real flight data and moves it. Lastly, the true method is simply named for being exactly the same as the real flight data; this is what the aircraft in real life would see if placed at these positions. These methods are those that answer research question 1.

4.3.1 Polyfitting

This method divides the flight data into eight bins which represent eight different distances from the start of the runway. Each bin's size is one nautical mile which adds up to 8 nautical miles coverage. For each bin, the mean and standard deviation is calculated to form a basis for constructing a Normal Distribution. Hence, this method generates flight data from normal distribution.

A continuous function is created of both the mean and standard deviation from the eight bins by polynomial regression. The input to the function is the distance from the runway; this polynomial regression is calculated through a package in the Python library Scipy. This implementation simplifies generating new functions for new flight data.

For lateral positioning flight data, a cubic expression produced the least amount of error for the mean and the standard deviation. The altitude flight data is linear; therefore, a first-order function produces the least amount of error. This linear relationship works for the mean in altitude data, not for the standard deviation. For the standard deviation of altitude, a third-order function was used. However, there is a significant error as the standard deviation varies and thus a low fit of the model, as is visible in Figure 12b.



(a) Lateral position flight data fit with polynomial regression.

(b) Altitude flight data fit with linear and polynomial regression.

Figure 12: Example of polynomial regression on flight data. The dotted lines are the fitted functions from polynomial regression. The points are the real data calculated from 8 bins of the flight data.

From the mean and standard deviation functions, data can be generated. First, a distance from the runway is randomly uniformly sampled, which allows for calculating the mean and standard deviation from that distance. The produced mean, and standard deviation is applied in a normal distribution to generate a position. This is the generated flight data. An example of regression performed on one set of flight data is visible in Figure 12.

4.3.2 Cube Normal

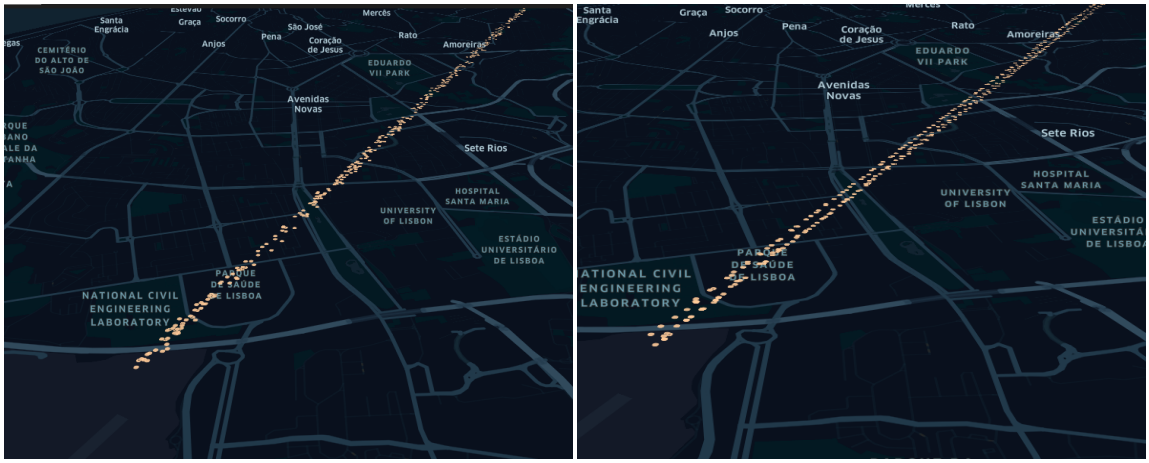
Similar to the polyfitt method in Section 4.3.1, this method divides the flight data into eight bins where one bin equals one nautical mile from the runway; hence the range of data is 0-8 NM from the start of the runway. The number of images generated is equally distributed for all bins. Altitude is

determined by the same normal distribution used for polyfitt. Lateral position is determined by a normal distribution created for each bin according to real data. This method, compared to polyfitt, focus more on specific sampling in each bin rather than using a continuous function between all data. An example of the sampling can be seen in Figure 13.

Figure 13: This figure shows a part of the cube normal distribution. The yellow dots are the sampled aircraft position according to the distribution. At the centre of the image are two bins divided. On each side, different lateral normal distributions are used for each bin. This results in the yellow dots being more spread out on the left side compared to the right. The image is captured in the geospatial analysis tool, Kepler.gl.

4.3.3 One Move

The One move distribution positions the aircraft according to a real flight position from the flight data and modifies it. This modification is a uniformly distributed shift of position 0-20 metres sideways and 0-20 metres up/downwards. This combination of random modification to the real data allows it to replicate the patterns from the real data while still being unbiased towards it. An example of this distribution can be seen in Figure 14a compared to the real flight data in Figure 14b.



(a) 1000 One move distribution samples according to the real flight data.

(b) 1000 real flight data measurements sampled in xplane.

Figure 14: These figures show the One move distribution and real flight data based on LPPT airports landing scenarios. The yellow dots are the aircraft position. The image is captured in the geospatial analysis tool, Kepler.gl.

4.4 Dataset Generation System

For an accurate representation of flight conditions and environments, X-Plane [21] is the simulated environment of choice. The programming language Scenic [23], VerifAI [67] and the X-plane plugins XPC and [28] will be used to manipulate the X-Plane environment and generate the dataset. The dataset generation process will be created according to a model, which can be seen in Figure 15. An accurate representation of landing approaches is given through the data distribution analysis and the landing approach data. This data is used to create Scenic scenarios which replicate the landing approach in the simulator. VerifAI is used to configure runway files and control X-Plane through the XPC plugins.

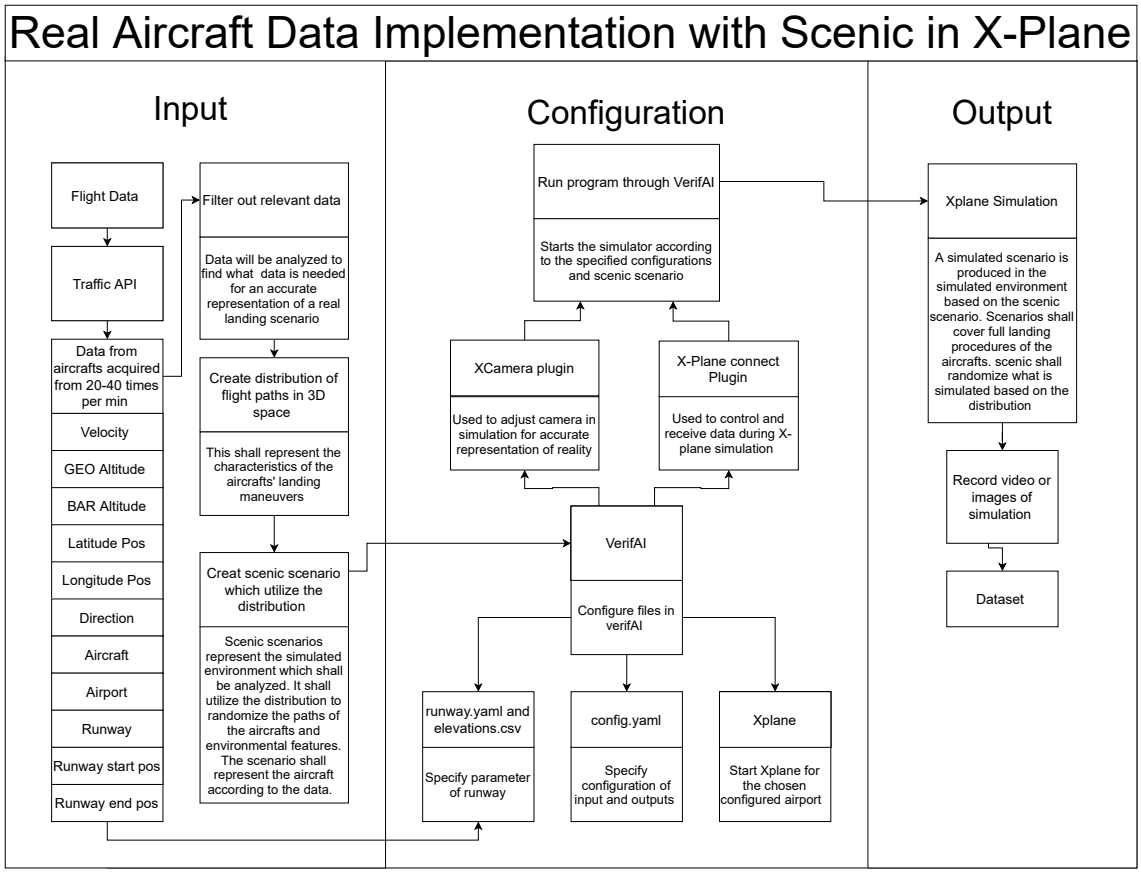


Figure 15: This model shows the proposed system, which utilises real flight data to generate scenarios in the simulator. The input part creates Scenic scenarios based on flight data and a distribution analysis. The configuration converts the Scenic scenario to a representation of the scenario in the simulator. Once represented, datasets can be gathered from it.

4.4.1 Scenic Modifications

The current iteration for using Scenic to control the X-Plane simulator only generate scenarios where the aircraft is on the runway and can be seen in Figure 16. Thus, modifications of the language are necessary to represent landing conditions. The current variables which can be manipulated to create scenarios are position, heading, aircraft and weather. The airport can also be chosen but requires additional information about the terrain. To represent real landing scenarios; the variables altitude, pitch, roll and camera angle needs to be added in the generation process.

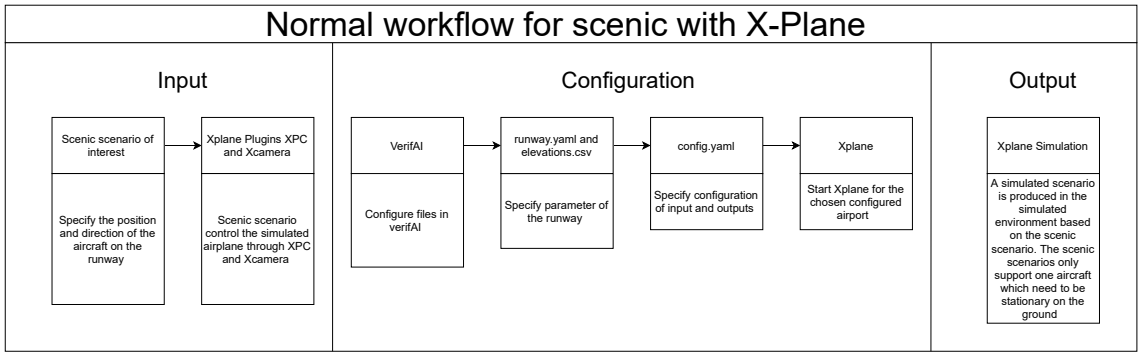


Figure 16: This model describes the original iteration of Scenic with X-Plane. A Scenic scenario is created through programming and applied to the simulator with VerifAI and X-Plane plugin. The generated scenarios can only simulate scenarios on the ground.

4.4.2 Balanced Distribution

Dataset generated through Scenic needs to represent a balanced distribution of landing scenarios. The dataset is considered balanced when it covers the majority of landing approaches that can occur. Thus, the system needs to generate datasets which consider different condition during landing approaches. The conditions considered are environmental conditions, position, attitude and different airports runways. The distribution analysis performed on the landing data will define the boundaries of these conditions. Conditions need to be represented in the simulator and in different combinations to resemble most possible landing approaches. Combinations of conditions can affect the detection different compared to separately. For example, nighttime and rain are conditions that affect each other. A dataset covering all combinations would result in huge amounts of data and require a lot of time to simulate. To limit these factors, the generation of datasets will first be performed with low amounts or no combinations of conditions. A CNN can then be applied to test the conditions affect on identifying runways. By using this approach, the importance of each condition can be recognised and utilised to balance the dataset distribution. If a condition heavily affects the detection process, boosting can be used to add more scenarios for a reliable dataset. If the condition doesn't affect the detection process, less images of that situation are needed.

4.4.3 X-Plane Experiment Configurations

During the generation of images in the simulator, graphics quality affects the world's aesthetics, amount of objects, and detail in the environment. To achieve more realistic images and provide more detail from which the CNN can learn from, graphics-related settings are maxed out. The settings used for all experiments can be seen in Appendix A.

For each of the RQs 1, 2, 3 and 4, the number of datasets needed and the simulation configuration changes. To analyse the effect of the variations separately, the other variations are set equally among the datasets. For RQ1, four datasets are produced based on the distributions One Move, Cube Normal, Polyfitting and real flight data. Two evaluation dataset are also created based on new real flight data and a new one move generation based on this data. The configurations for the scenarios and datasets produced for RQ1 can be seen in Table 2.

DataSet ID	Positioning Method	Pitch (deg)	Yaw(deg)	Roll (deg)	Airport	Time	Clouds (0-6)	Rain (%)	Images
Dataset True (Training)	Real Flight Data From Set LPPT 1	0	Accoring to runway	0	LPPT	14:00	0	0	8000
Dataset OneMove (Training)	One Move From Set LPPT 1	0	Accoring to runway	0	LPPT	14:00	0	0	8000
Dataset Cube Normal (Training)	Cube Normal	0	Accoring to runway	0	LPPT	14:00	0	0	8000
Dataset Poly (Training)	Polyfitting	0	Accoring to runway	0	LPPT	14:00	0	0	8000
Dataset True (Evaluation)	Real Flight Data From Set LPPT 2	0	Accoring to runway	0	LPPT	14:00	0	0	8000
Dataset One Move (Evaluation)	One Move From Set LPPT 2	0	Accoring to runway	0	LPPT	14:00	0	0	8000

Table 2: This table describes the datasets generated and their configuration to analyse the effect of different distribution algorithms used for positioning. Each row represents a dataset generated, and all variables except "Dataset ID" determine the scene in X-Plane according to their values.

For RQ2, the change of airport/runway is analysed by generating two training sets and two validation datasets for comparison. The evaluation set "Dataset New True" is the same set generated for training but used as a baseline evaluation. This generation is created according to real data from their respective airport. The configurations for the scenarios and datasets produced can be seen in Table 3.

DataSet ID	Positioning Method	Pitch (deg)	Yaw(deg)	Roll (deg)	Airport	Time	Clouds (0-6)	Rain (%)	Images
Dataset New True (Training)	Real Flight Data From Set LPPT 3	0	Accoring to runway	0	LPPT	14:00	0	0	8000
Dataset OneMove Expanded (Training)	One Move From Set LPPT 3	0	Accoring to runway	0	LPPT	14:00	0	0	20000
Dataset New True (Evaluation)	Real Flight Data From Set LPPT 3	0	Accoring to runway	0	LPPT	14:00	0	0	8000
Dataset OTHH (Evaluation)	Real Flight Data From Set OTHH 1	0	Accoring to runway	0	OTHH	14:00	0	0	8000
Dataset LFPO (Evaluation)	Real Flight Data From Set LFPO 1	0	Accoring to runway	0	LFPO	14:00	0	0	8000

Table 3: This table describes the datasets generated and their configuration to analyse changing airport/runway. Each row represents a dataset generated, and all variables except "Dataset ID" determine the scene in X-Plane according to their values.

RQ3 aims to analyse environmental condition which might affect the functionality of the CNN. Three datasets are generated where each represents an environmental condition affecting the runway. Dataset Time adds a uniformly distributed time variation between 06:30 and 19:00 for each image generated. Dataset Clouds uniformly change the clouds according to the six cloud types(0-5) in X-Plane. Dataset Rain uniformly changes the rain levels in X-Plane according to a percent value. The reason for choosing these weather conditions is due to these being the most common ones and directly accessible in X-Plane. Other ones, such as snow, required texture packs to be utilised, and thunder was hard to capture during generation due to its unpredictability. The configurations for the scenarios and datasets produced can be seen in Table 4.

DataSet ID	Positioning Method	Pitch (deg)	Yaw(deg)	Roll (deg)	Airport	Time	Clouds (0-6)	Rain (%)	Images
Dataset Time (Evaluation)	Real Flight Data From Set LPPT 3	0	Accoring to runway	0	LPPT	6:30 to 19:00	0	0	8000
Dataset Clouds (Evaluation)	Real Flight Data From Set LPPT 3	0	Accoring to runway	0	LPPT	14:00	Uniformly Distributed (0-5)	0	8000
Dataset Rain (Evaluation)	Real Flight Data From Set LPPT 3	0	Accoring to runway	0	LPPT	14:00	0	Uniformly Distributed (0-100)	8000

Table 4: This table describes the datasets generated and their configuration to analyse the effect of the environmental conditions on the runway. Each row represents a dataset generated, and all variables except "Dataset ID" determine the scene in X-Plane according to their values.

The final generation performed, analyse attitude changes of the aircraft to answer RQ4. Three evaluation datasets are generated where each covers the aircraft attitudes yaw, pitch and roll. These parameters are augmented for each image according to a uniform distribution in a threshold realistic to the normal function 0f A320 aircraft during landing [68]. The configurations for the scenarios and datasets produced can be seen in Table 5.

DataSet ID	Positioning Method	Pitch (deg)	Yaw(deg)	Roll (deg)	Airport	Time	Clouds (0-6)	Rain (%)	Images
Dataset Pitch (Evaluation)	Real Flight Data From Set LPPT 3	(0-10)	Accoring to runway	0	LPPT	14:00	0	0	8000
Dataset Yaw (Evaluation)	Real Flight Data From Set LPPT 3	0	Accoring to runway +-(0-20)	0	LPPT	14:00	0	0	8000
Dataset Roll (Evaluation)	Real Flight Data From Set LPPT 3	0	Accoring to runway	+-(0-7)	LPPT	14:00	0	0	8000

Table 5: This table describes the datasets generated and their configuration to analyse the effect of aircraft attitude on detecting the runway. Each row represents a dataset generated, and all variables except "Dataset ID" determine the scene in X-Plane according to their values.

4.4.4 Annotations

Once a dataset is generated, each image needs to be annotated to mark features that the CNN shall detect. This procedure is usually done manually for captured images and is a very time-consuming process. Since the outcomes of the thesis require multiple datasets and trained CNNs, the annotation process needs to be automatic. This automation would require the generation system to know where the runway is on the generated picture and write it to an annotation file. Since the generation process occurs in the simulator, variables such as aircraft position, attitude and altitude are known. Runway position and elevation is also known in global coordinates, which directly translate into the simulator. With these variables available, three-dimensional (3D) camera projection can be used.

3D projection is a technique used to display a 3D object on a two-dimensional (2D) surface. In X-plane, the 3D space represents the global coordinate system, and the 2D surface is the image plane displayed on the monitor. The image plane depends on the camera, which is positioned on the aircraft during simulation. By transforming the 3D points on each runway corner to the image plane, their position on the screen is given.

Saab provided the code for this transformation through an earlier developed project in the company. The method used consists of multiple transformations and rotations to convert the world coordinates to camera coordinates and can be described in the following steps.

1. Input runway corners position, aircraft position and attitude.

$$\begin{aligned} Runwaypoint1(Lat, Long, Altitude) &= (38.76521, -9.144896, 30) \\ Runwaypoint2(Lat, Long, Altitude) &= (38.764896, -9.144391, 30) \\ Runwaypoint3(Lat, Long, Altitude) &= (38.797463, -9.127649, 30) \\ Runwaypoint4(Lat, Long, Altitude) &= (38.797266, -9.127034, 30) \\ Aircraftposition(Lat, Long, Altitude) &= (38.71357345581055, -9.172924995422363, 632) \\ Aircraftattitude(Heading, Pitch, Roll) &= (0.4978223326170012, 0.0, 0.0) \end{aligned} \tag{7}$$

2. Convert the lat, lon and elevation coordinate system for the runway and aircraft coordinates to a cartesian ECEF(earth-centred, earth-fixed) coordinate system represented in metres. The formulas for the conversions can be seen in Equation 8. x is the latitudinal position, y is the longitudinal position and z is the altitude.

$$\begin{aligned} a(EarthSpheroidSemiMajor) &= 6378137 \\ b(EarthSpheroidSemiMinor) &= 6356752.3142 \\ x &= (a/\sqrt{\cos lat^2 + (b/a)^2 * \sin lat^2} + alt) * \cos lat * \cos lon \\ y &= (a/\sqrt{\cos lat^2 + (b/a)^2 * \sin lat^2} + alt) * \cos lat * \sin lon \\ z &= (b/\sqrt{(b/a)^2 * \cos lat^2 + \sin lat^2} + alt) * \sin lat \end{aligned} \tag{8}$$

3. Invert the attitude and position of the aircraft coordinates to move the camera instead of the world.

4. Calculate the camera to world translation, ECEF to ENU(East, North, Up) rotation and ENU to Tait Bryan rotation matrices. Camera to world translation matrix are used to convert camera coordinates to world coordinates and can be seen in Matrix 9. x is the latitudinal position in metres, y is the longitudinal position in metres and z is the altitude in metres. The rotation matrices rotate the attitude from ECEF to Tait Bryan angles and utilize standard rotation matrices 10. θ is the rotation angle and x, y and z represent rotation around that axis. Tait Bryan angles are a form of Euler angles used in aviation to describe the rotation of aircraft attitude.

$$\begin{bmatrix} 1 & 0 & 0 & -Aircraft(x) \\ 0 & 1 & 0 & -Aircraft(y) \\ 0 & 0 & 1 & -Aircraft(z) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x & 0 \\ 0 & -\sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0 & 0 \\ -\sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

5. Calculate the matrix multiplication for step 4 translation and rotations to receive the transformed runway points.

6. Project the transformed runway point according to distance to the image plane to receive camera coordinates for the runway points. This projection is done by switching to camera coordinates for each runway point and then use Equation 11 for the x and y values(the first two rows) in the matrix.

$$CameraCord = \frac{PointInMatrix}{distance * FocalLength} \tag{11}$$

4.4.5 Dataset Validation

To validate the datasets ability to represent landing approaches, a CNN will be trained on it and tested against different landing approaches. The dataset will be validated according to the CNN performance in detecting runways for different situations in the simulator. Same CNN will be used for all testing to minimise accuracy error from the network. Since the annotation system is automatic, annotations written need to be validated as well. For this validation, a sample of 40 images was analysed to see if there annotation was correct according to the image.

4.5 Object Detection Network Training Setup

Detectron2 is used as a platform for training the object detection model. The reason for deciding to use Detectron2 over an alternative is because it is well established and easy to use. It is integrated with GPU support and allows for access to backbone models through the Detectron2 model zoo. Furthermore, research made in parallel to this at SAAB also applied Detectron2. This allowed for collaboration and aid if possible when using Detectron2.

The CNN Faster R-CNN is used because it is an object detection architecture, and it is accessible from Detectron2 model zoo.

4.5.1 Determining the Backbone Model

For determining the backbone model for object detection, an experiment was conducted. The experiment consisted of training with several different backbone models available in the Detectron2 model zoo. The setup for the experiment consisted of generating images with annotated runways present, and training on the images with the backbones and lastly, producing metrics by using COCO evaluation. There are also metrics generated, such as the time it took to train and Frames-Per-Secound (FPS) of running the trained model. However, the main metrics used to evaluate the backbone is the AP score from the COCO evaluation.

The backbone models tested are:

- ResNet-50-FPN-3x
- ResNet-101-FPN-3x
- ResNeXt-101-32x8d-FPN-3x
- ResNet-50-DC5-3x

These backbone models were chosen as they had high AP as described in the Detectron 2 model zoo. Furthermore, the choices are also motivated on the basis of the results in [61].

During training, the setup used in Detectron2 consisted of using 455 annotated images of runways at a distance from 0 to 8 NM. Where the method used for generating the images were one move as described in Section 4.3.3. The iterations parameter was set at 10000. The batch size was 128, and training was performed on the graphics card Geforce 3090. The images are produced in the Xplane simulator and contain the Humberto Delgado Airport in Lisbon, runway 03.

4.5.2 Final Training Setup

The final training setup consisted of applying Detectron2 together with the backbone model ResNet-50-FPN-3x. This setup is used for answering research questions 1, 2, 3, 4. All the training runs were performed on an Nvidia Geforce 3090. The configuration used during the training was as follows:

- 90000 Iterations
- 128 Batch size
- 0.0025 Learning rate
- One class (Runway)
- 8000 Images

The reason for using these parameters for training was because they were recommended by a person working in parallel at SAAB, which had more experience in Detectron2.

For one larger set of flight data, generated by the one move method with 20000 images, the following setup was instead used:

- 200000 Iterations
- 128 Batch size
- 0.0025 Learning rate
- One Class (Runway)
- 20000 Images

4.6 Evaluation of the Object Detection

For all the research questions except the last, prediction accuracy is necessary. This prediction accuracy is the AP-score generated from evaluating images by an object detection model. A description of how this score is calculated is located in Section 2.9.

There are five types of AP scores used during the evaluation. These are AP, AP50, AP75, APs, APm and APl. Having all of these AP metrics allows for more interpretation of a models performance. Just the "AP" metric is the primary metric, as it covers an extensive range of different IoU, which gives a summary of the whole performance. AP50 and AP75 give more information about the model performance at just one specific IoU-threshold. The metrics APs, APm and APl all give information on how the object detection models perform depending on the size of the runway in the images. Therefore, if a runway is large inside an image, the APl will be changed during an evaluation. Similarly, if the runway is small, this will affect the APs.

The size of the runway in an image is dependant on the distance to the runway. Therefore, the APs, APm and APl are useful in determining an object detection models performance depending on the distance to the runway.

For research questions 1, 2, 3 and 4, the AP is used for determining the prediction accuracy. AP is used for measuring the prediction accuracy since it is well established in evaluating the performance of object detection models. The mAP or AP in the case of single classification is the de-facto standard way of evaluating regression and classification task such as object detection.

For all the setups used to answer the research questions, the AP-score is generated by COCO evaluation 2.9.1. The reason for choosing COCO evaluation is that it can evaluate object detection, and it is implemented in Detectron2 for easy use.

5. Ethical and Societal Considerations

The methods applied in this paper for conducting the research have no interaction with human subjects or elements.

Using an object detection for finding and localising runways in real landing approaches can aid in reducing fuel emissions. Reducing fuel emissions are necessary for minimising the progress of global warming. Furthermore, this reduction may increase air quality and diminish costs.

An object detection model that aid in finding and localising the runway present in images may allow for advances in autonomous landings. Autonomous landings are more optimised in their landings and thus more efficient. This efficiency reduces the amount of fuel emissions by reducing unnecessary time in the air. Unnecessary time can occur when an aircraft has to perform a go-around when a safe landing can not be made, or an aircraft has to perform a holding pattern to wait for their window of time for landing. Adding systems that increase awareness by knowing the location of the runway may aid in preventing these unnecessary time generating conditions. The system which applies this object detection can increase safety during landing.

A poorly trained network for detecting runways may cause erroneous placement of the runway, imposing a possible hazard that can put humans at risk. However, the means for creating more reliable object detection models may be possible by the methods applied in this paper. Thus, limiting erroneous scenarios, i.e. false positives or false negatives, which allows for safe implementation.

A complement to generating images of runways in a simulator is doing it from a real aircraft by placing a camera that records the landing approach. This would take fuel due to special flight approaches that need to be covered and thus not good for the climate.

6. Results

In this section the results from experiments and analyses is presented.

6.1 Landing Approach Flight Data

Table 6 represents flight data received from the OpenSky Network Database.

ICAO of airport	Start Timestamp (UTC)	End Timestamp (UTC)	Size
OTHH	2021-01-01 15:00:00+0000	2021-01-15 16:30:00+0000	22709282
SBVT	2021-01-01 15:25:00+00:00	2021-01-31 16:30:00+00:00	2859902
LSZH	2019-01-01 15:00:00+00:00	2019-01-15 16:30:00+00:00	26656601
EHAM	2019-05-01 15:00:00+00:00	2019-05-12 16:30:00+00:00	47155142
(1) LPPT	2021-01-01 15:00:00+00:00	2021-01-29 16:30:00+00:00	11427991
(2) LPPT	2021-02-01 15:00:00+00:00	2021-02-28 16:30:00+00:00	6026162
(3) LPPT	2021-03-01 15:00:00+00:00	2021-04-28 16:30:00+00:00	21690299
EGLF	2021-01-01 15:00:00+00:00	2021-01-29 16:30:00+00:00	2193742
LFBO	2018-02-01 09:00:00+00:00	2018-02-28 16:00:00+00:00	11733303
LFPO	2021-01-01 15:00:00+00:00	2021-01-29 16:30:00+00:00	17067034
ELLX	2021-01-01 15:00:00+00:00	2021-01-29 16:30:00+00:00	9672354

Table 6: ADS-B datasets queried from the OpenSky Network. Size is the amount of ADS-B points present in the data.

Table 7 is the outcome of the filtering performed on the data in Figure 6. Most of the datasets are of aircraft type A320, as it was the most prevalent in the flight data. Similarly, the runway is chosen based on the size of available flight data for aircraft landing at that runway. Notably, the size of the datasets after filtration is significantly less than what was received from the OpenSky Network database.

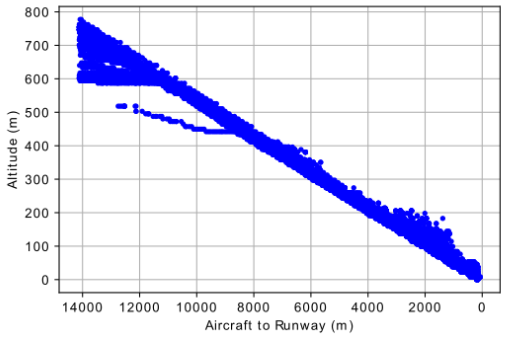
After filtering the flight data for only landing approach, as described in Section 4.1, the resulting data shall only contain valid landing approach data. Therefore, the outcome of this filtered data is analysed visually through graphs. These graphs can be seen in Figure 17.

What can be observed in Figure 17b is that many flights are connecting from one side. There are some points that may be erroneous, indicated by the red dots between 8000 to 1000 metres. However, it seems only to be around 4 of these points. Seemingly no flight trajectory exists in the flight data, which has no intention to land, as would be indicated by a series of points crossing over perpendicularly or not converging around the x-axis. Once the aircraft has reached 8000 m from the airport, they are mostly set for lateral positioning in the approach.

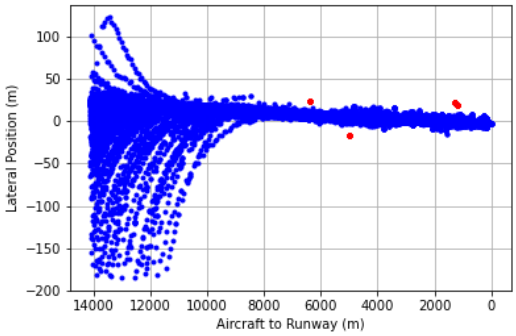
The altitude flight data in Figure 17a does not seem to have any erroneous data. The only noticeable from this data is the flight that connects to the landing approach later and at a lower altitude at around 8000 metres.

Set	ICAO of Airport	Runway	Aircraft typecode	Size
1	OTHH	34R	A320	53462
2	SBVT	02	B738	11832
3	LSZH	14	A320	1485
4	EHAM	06	B738	3378
5	(1) LPPT	03	A320	21521
6	(2) LPPT	03	A320	4165
7	(3) LPPT	03	A320	24288
8	LFBO	32L	A320	9833
9	LFPO	25	A333	10250

Table 7: Various flight datasets generated from downloading and filtering data for only landing approach. Size is the amount of ADS-B points present in the data.



(a) Altitude flight data.



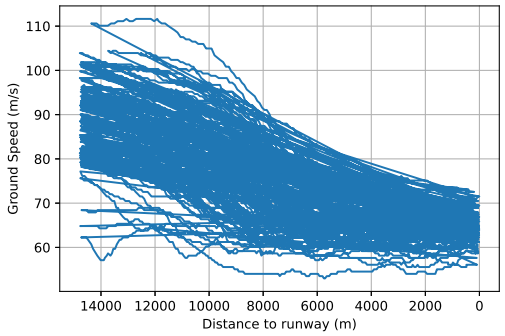
(b) Lateral positioning flight data.

Figure 17: Landing approach flight data at Hamad International Airport runway 34R, all Airbus A320. Each blue point represents one ADS-B data. A connected series of these points is one flight trajectory. There are 285 unique landing trajectories in this flight data. The data present in the graph belongs to set 1 in Table 7. The red dots in 17b resembles erroneous data.

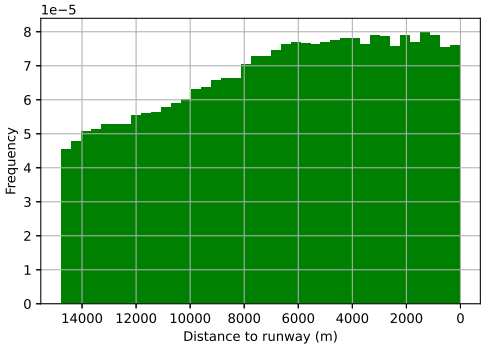
6.2 Distribution Analysis of Real Flight Data

Following the generation of landing approaches, the distribution of data was necessary for analysis. In this section are the distributions of the three parameters necessary for replication of the flight data. These three parameters being the aircraft to runway distance, lateral positioning and altitude.

The distribution for aircraft to runway is mostly uniform, as seen in Figure 18b, as one flight coming in for landing will be located in all ranges; however, as aircraft lowers their speed when closer to the runway, there are more flight data due to the extended time present in that range. The lowering of the speed is visible in the ground speed as seen in Figure 18a.



(a) Ground speed.



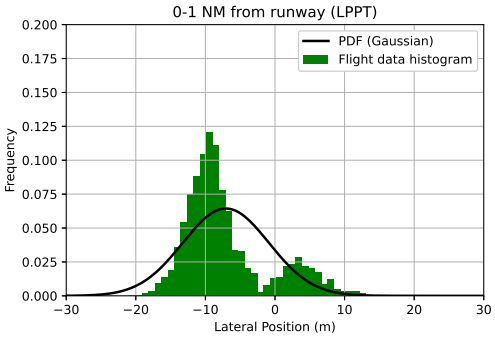
(b) Histogram of ADS-B point density distribution over distance to runway.

Figure 18: Flight data of aircraft to runway and ground speed at landing approach. The configuration for the dataset is seen in Table 7, set 5.

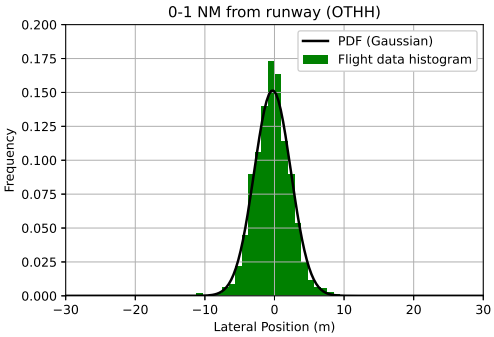
The distribution for the lateral position is seen in Figure 19. Here are the distributions displayed together with the corresponding Probability Distribution Function (PDF) for normal or Gaussian. This PDF is used when applying the methods that replicate flight data, such as the polynomial regression method or the cube normal method. This analysis is also necessary for understanding how the methods will perform against the real flight data.

The distribution from lateral positioning are not necessarily similar from airport to airport, as is seen in Figure 19a and 19b. The first takes a multimodal distribution while the second resembles a normal distribution, which is unimodal. The fact that the distribution in Figure 19b is normally distributed is also supported by the histogram being almost fully contained by the PDF function.

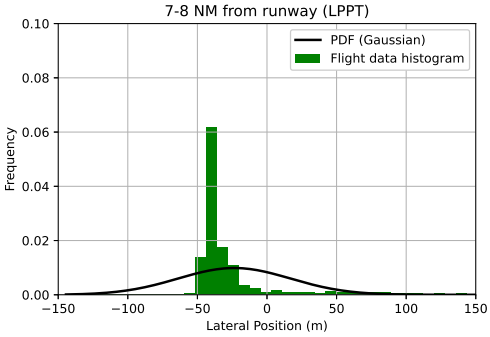
At distances further from the runway, the data lacks in resembling normal distribution, as is seen in the Figures 19c and 19d. The distribution is asymmetric, mostly due to data from aircraft turning into the runways centerline bearing, i.e. from base leg to final. The Figure 1, visualise what the base leg to final is. This asymmetry is not prevalent in the distributions closer to the runway. The spread of the normal distribution plot represented by the PDF is high in both 19c and 19d. Higher than the real lateral flight data. This is because of the flight data from aircraft coming in from base leg to final for the landing approach.



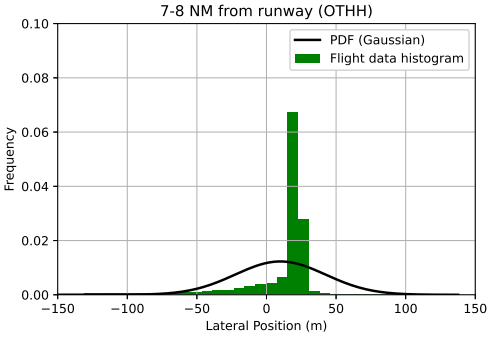
(a) LPPT



(b) OTHH



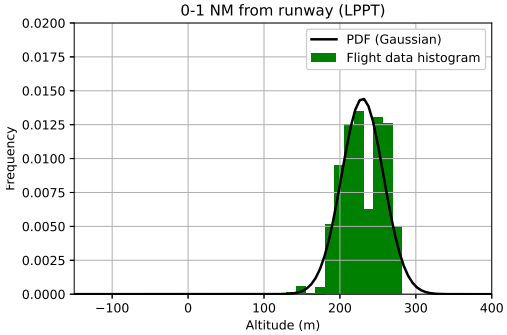
(c) LPPT



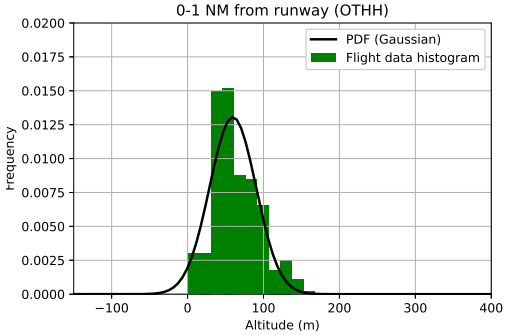
(d) OTHH

Figure 19: Distributions of the lateral positioning. The green histogram represents the real data; the black line represents the normal distribution fit to the real data. Note axis range difference between the 0-1NM and 7-8 NM. The data present in the graph belongs to set 5 in Table 7.

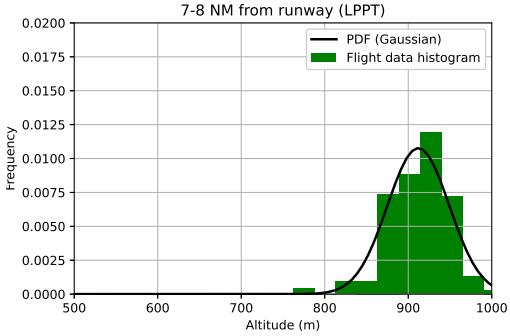
As for the altitude distributions, some has a resemblance to normal distributions. As most aircraft landing approach aims at a 3-degree angle when landing, this is mostly followed except when influenced to some variation. The 3-degree angle of the landing approach is visible in Figure 21. However, aircraft altitude data is also multimodal in some cases, as seen in Figure 20d.



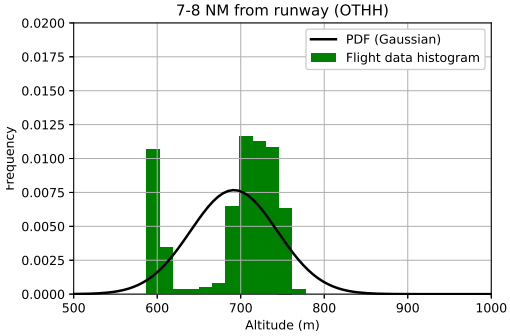
(a) LPPT



(b) OTHH



(c) LPPT



(d) OTHH

Figure 20: Distributions of the altitude. The green histogram represents the real data; the black line represents the normal distribution fit to the real data. Note axis range difference between the 0-1NM and 7-8 NM. The data present in the graph belongs to set 5 in Table 7.

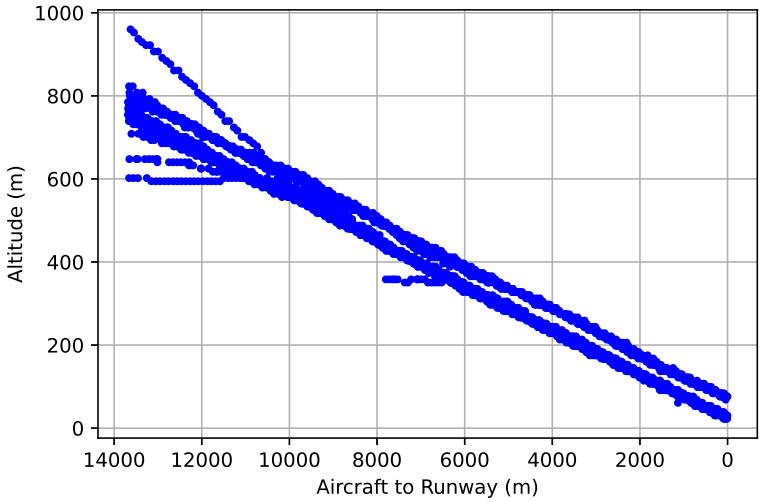


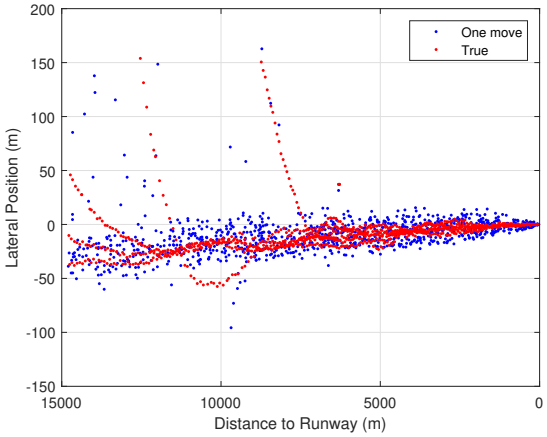
Figure 21: Landing approach altitude flight data at Amsterdam Airport Schiphol, EHAM. This data correspond to set 4 in Table 7.

6.3 Comparing True and Generated Flight Data

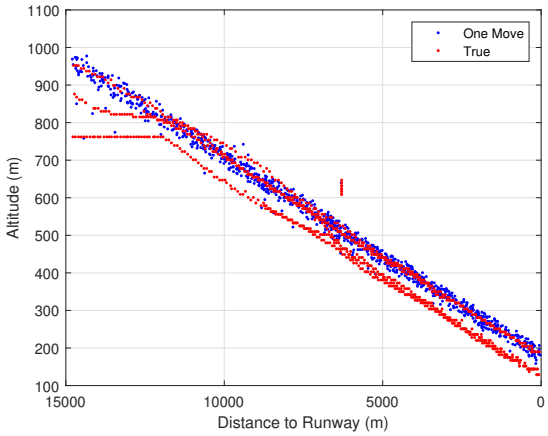
After the generated images have been produced using the methods one move, polyfit and cube normal described in Section 4.3, the result can be compared against the true flight data. The true and generated flight data for altitude and lateral position is visible in Figure 22.

For lateral positioning, the one move method has a spread which coincides with the true flight data at distances further away from the runway. This is because it captures the flight data where aircraft transition from base leg to final approach. This can be seen in Figure 22a, where the generated points by one move are visible by the blue dots further away from the x-axis, above around 15 in the y-axis. The methods that apply normal distribution, i.e. polyfit and cube normal, are limited in capturing the base-to-final turn data in lateral position, as shown in Figures 22c and 22e. The distribution is more spread due to this base-to-final data. For lateral position, the one move method has a higher spread than the true flight data closer to the runway; this is visible from the range 0 to 5000 metres in Figure 22a. Compared to the two other methods, which spread resembles that of the true method.

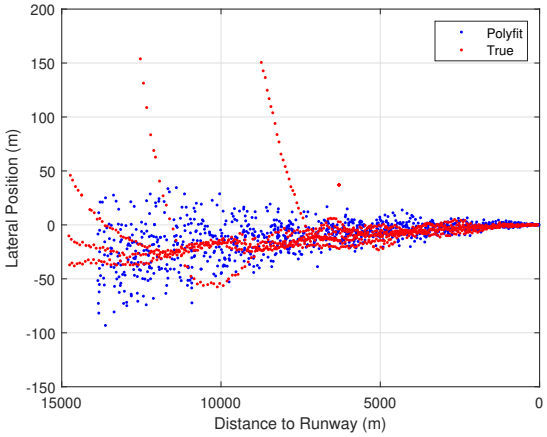
The altitude is less spread for the one move method in Figure 22b compared to the other. The polyfit and cube normal method do not generate a point from the erroneous altitude data from true flight data present around 6000 m from runway and above 600 m altitude. However, they are affected as the erroneous data will change the mean and spread, applying a minor shift. The one move method does produce a point at the erroneous data, as visible in Figure 22b.



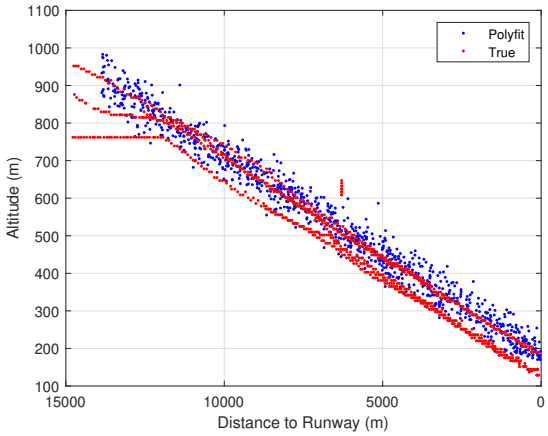
(a) Lateral Position One Move method



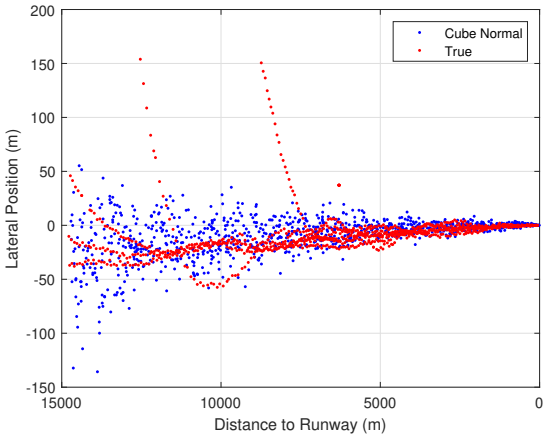
(b) Altitude One Move method



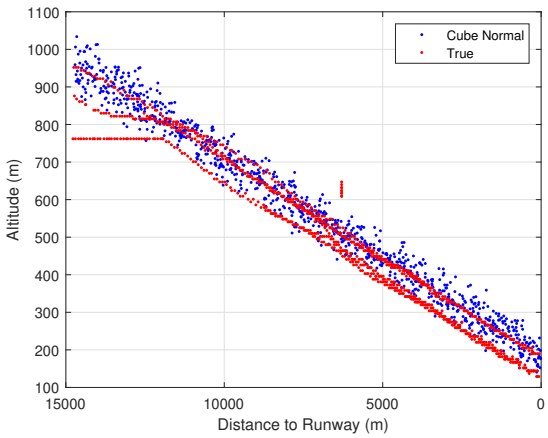
(c) Lateral Position Polynomial Regression method



(d) Altitude Polynomial Regression method



(e) Lateral Position Cube Normal method



(f) Altitude Cube Normal method

Figure 22: Scatterplots of true flight data from ADS-B (red) and generated flight data (blue) based on three methods. The configuration for the true flight data is visible in Table 7, set 5.

6.4 Experiment Results for Determining the Backbone Model

Here is presented the results from the experiment as described in Section 4.5.1. A description for the metrics present in Table 8 is available in Section 2.9.

What can be seen in Table 8, is that for the primary challenge metric (AP), the backbone model

ResNet-50-FPN-3x has the highest value. This AP is the averaged AP over several IoU thresholds, which is exclusive to COCO evaluation. It is the most significant metric in this data. The second highest for this metric is ResNeXt-101-32x8d-FPN-3x, with only a 0.412 difference between the highest. The last two backbone models have relatively low AP in comparison to the two highest.

Besides the COCO AP metric, the backbone model ResNeXt-101-32x8d-FPN-3x outperforms ResNet-50-FPN-3x in AP50, AP75, APm and APl. However, the difference is minor. As for training time and FPS, the backbone model ResNet-50-FPN-3x has a better performance than ResNeXt-101-32x8d-FPN-3x. Although these two metrics are not major in terms of significance, the difference is so large that they aid in concluding which backbone model has the best performance when there is a relatively low distinction between the two models.

Backbone Model (Faster R-CNN)	**AP**	**AP50**	**AP75**	**APs**	**APm**	**APl**	**Training time**	**FPS**
ResNet-50-FPN-3x	70.987	98.820	84.919	66.347	85.635	88.482	01:21:17	27.30
ResNet-101-FPN-3x	66.297	98.907	82.837	62.548	72.278	84.512	01:27:58	14.91
ResNeXt-101-32x8d-FPN-3x	70.575	98.888	87.010	65.647	85.846	88.611	01:54:02	7.00
ResNet-50-DC5-3x	62.259	97.781	69.512	53.726	77.695	84.255	01:14:09	4.00

Table 8: Table demonstrating the results from running several backbone models with the corresponding Average Precision. The FPS corresponds to the inference FPS.

6.5 Object Detection Evaluation

This section refers to the evaluation of running object detection models on sets of images. The result of this being the AP-score which comes from using the COCO evaluation in Detectron2 and images containing predictions.

6.5.1 Flight Data Replication Methods

The result of this section refers to research question 1. The process for creating this is found in Section 4.0.2. Tables 9 and 10 show the results from evaluating 4 object detection models each produced with different methods. These methods being true, one move, poly and cube normal as described in Section 4.3. One image set used for the evaluation was created by the true method with flight data from the same airport and runway. The other image set for evaluation is generated by the one move method. The Diagram 6 shows how these two evaluating image sets were produced, named "Images New Set" and "Images Large New Set".

Notably, for the primary AP-score (column 2), the true model generates the highest in Table 9 with an AP-score of 64.129. As for the second highest the cube normal method is close with 63.296. As for the latter, they have a similar score with almost 5 points less than the two other methods.

For Table 10, cube normal has the highest AP of 47.185.

Model	**AP**	**AP50**	**AP75**	**APs**	**APm**	**APl**
True	64.129	97.684	66.742	51.954	87.742	95.154
One Move	58.557	97.467	55.288	41.535	84.713	97.954
Poly	58.186	95.461	60.204	44.090	79.984	90.442
Cube Normal	63.296	97.598	67.372	50.519	79.688	89.899

Table 9: AP scores from running 4 object detection models all trained on images produced by different methods. The evaluation image set for this data is generated with flight data that belongs to set 6 in Table 7. This evaluation image set corresponds to "Images New Set" in Figure 6.

Model	AP	AP50	AP75	APs	APm	APl
True	43.286	79.866	40.168	23.670	84.717	87.722
One Move	44.395	81.009	40.320	21.044	86.800	91.499
Poly	43.001	78.440	40.377	25.197	81.261	85.575
Cube Normal	47.185	85.663	43.385	28.070	81.660	87.123

Table 10: AP scores from running 4 object detection models all trained on images produced by different methods. The image set for this evaluation corresponds to 20000 images from set 7 in Table 7. This evaluation image set corresponds to "Images Large New Set" in Figure 6.

6.5.2 Changing Runway

The result in Tables 11, 12 and 13 provide answers to research question 2. The object detection models for this case is trained on images from LPPT runway 03. The image set used in Table 11 for evaluation is also LPPT runway 03. One Move Expanded and New True models in the table refers to "Model 1 Trained on Airport 1" and "Model 2 Trained on Airport 1" under "Evaluating object detection model" seen in the Diagram 7.

Model	AP	AP50	AP75	APs	APm	APl
One Move Expanded	60.242	98.439	58.824	45.248	83.425	94.835
New True	62.877	97.779	64.069	48.677	88.183	93.632

Table 11: AP scores from running 2 object detection models on a image set generated by the true method. The image set for this evaluation corresponds to 8000 images at Humberto Delgado Airport, Lisbon, LPPT.

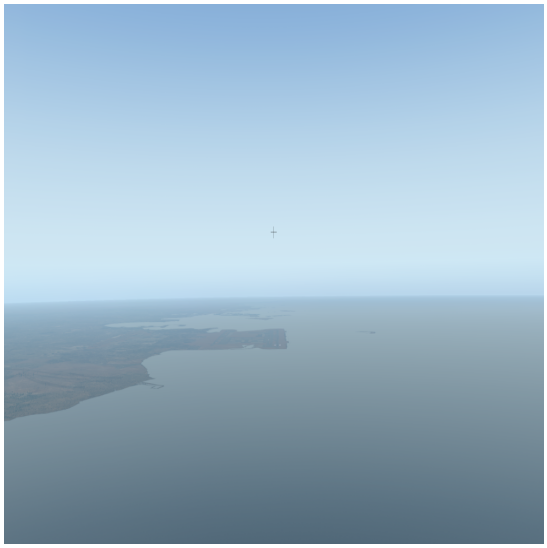
Model	AP	AP50	AP75	APs	APm	APl
One Move Expanded	4.708	14.796	0.052	0.000	7.753	28.590
New True	5.587	16.659	0.050	0.000	9.954	31.707

Table 12: AP scores from running 2 object detection models on a image set generated by the true method. The image set for this evaluation corresponds to 8000 images at the Hamad International Airport, OTHH.

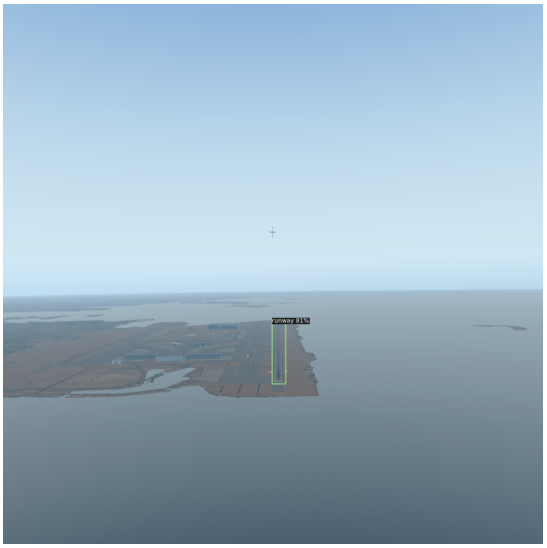
Model	AP	AP50	AP75	APs	APm	APl
One Move Expanded	23.208	64.600	9.495	12.143	49.432	50.684
New True	30.294	79.392	11.257	22.465	51.434	47.953

Table 13: AP scores from running 2 object detection models on a image set generated by the true method. The image set for this evaluation corresponds to 8000 images at the Paris-Orly Airport, LFPO.

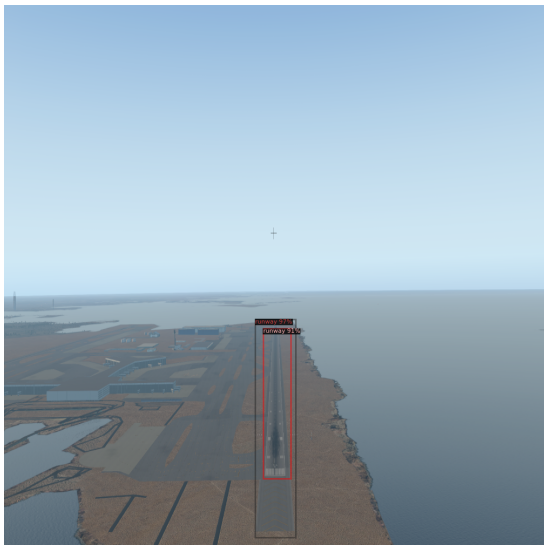
Below, the predictions of the "New True" object detection models are displayed. These predictions are made on images of an landing approach at Hamad International Airport 23 and Paris Orly Airport 24.



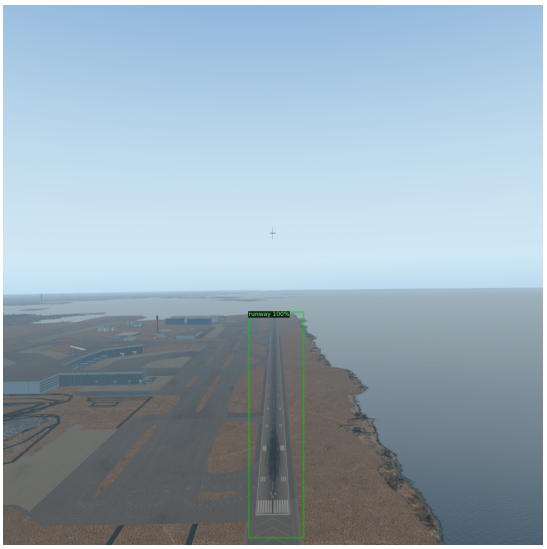
(a) Furthest Distance (13387 metres to runway)



(b) First Detection (3062 metres to runway)

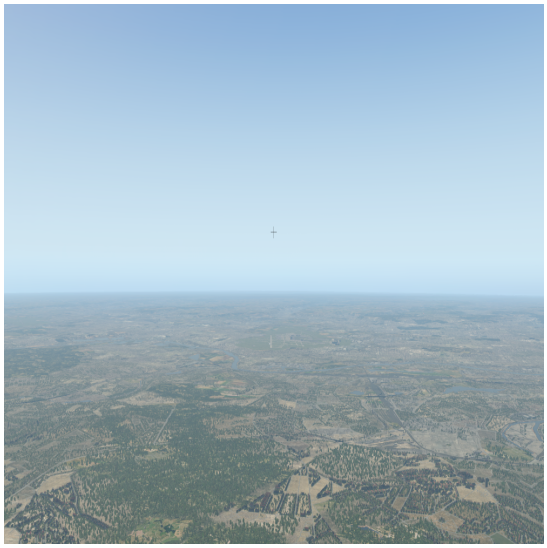


(c) Redundant Prediction (1084 metres to runway)

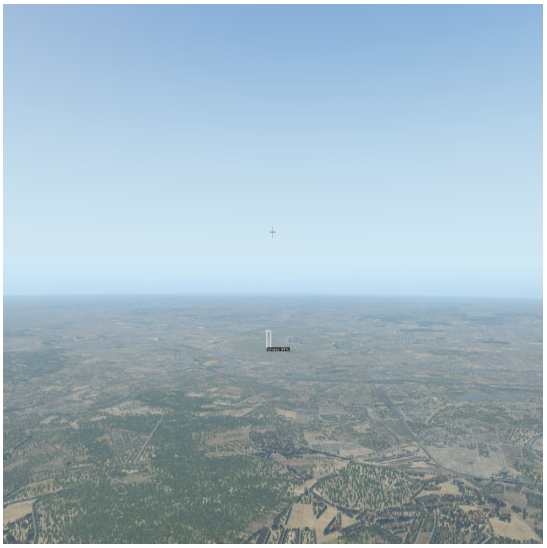


(d) Closest to runway (526 metres to runway)

Figure 23: Landing approach at Hamad International Airport runway 32R. Images are generated in Xplane. The boxes are predictions made by object detection models. The images are from one trajectory; thus, the furthest distance in Figure 23a, is not necessary the furthest away from the runway over all the flight data in the set.



(a) Furthest Distance (14890 metres to runway)



(b) First Detection (14093 metres to runway)

(c) Close to runway (5039 metres to runway)

(d) Closest to runway (718 metres to runway)

Figure 24: Landing approach at Paris Orly Airport runway 25. Images are generated in Xplane. The boxes are predictions made by object detection models. The images are from one trajectory; thus, the furthest distance in Figure 24a, is not necessary the furthest away from the runway over all the flight data in the set.

6.5.3 Environmental Conditions

The results present in Tables 14 and 15 are of to 2 environmental conditions in the research question 3. The first being the clouds and the second being the time of day. The prediction accuracy for this is similarly calculated to the previous research question, by evaluating object detection models on images, that then generates AP. One Move Expanded and New True in the table refers to the two object detection models described in the Diagram 8.

The rain environmental condition is not included in the data.

Model	**AP**	**AP50**	**AP75**	**APs**	**APm**	**APl**
One Move Expanded	45.493	79.747	42.401	28.197	79.159	92.562
New True	51.763	80.635	53.044	34.813	85.708	93.047

Table 14: AP scores from running 2 object detection models on a image set generated by the true method. The image set for this evaluation corresponds to 8000 images at Humberto Delgado Airport, Lisbon, LPPT, where there are clouds present.

Model	**AP**	**AP50**	**AP75**	**APs**	**APm**	**APl**
One Move Expanded	51.853	89.235	47.797	37.228	75.790	89.639
New True	55.062	92.241	52.063	40.890	81.713	89.880

Table 15: AP scores from running 2 object detection models on a image set generated by the true method. The image set for this evaluation corresponds to 8000 images at Humberto Delgado Airport, Lisbon, LPPT, with changing time.

Below, the predictions of the "New True" object detection models are displayed. These predictions are made on images where there are a varying amount of clouds and changing time of day. The variation in clouds are visible in column two in Figure 25; for the same figure, the changing times that affect the images are visible in column three.

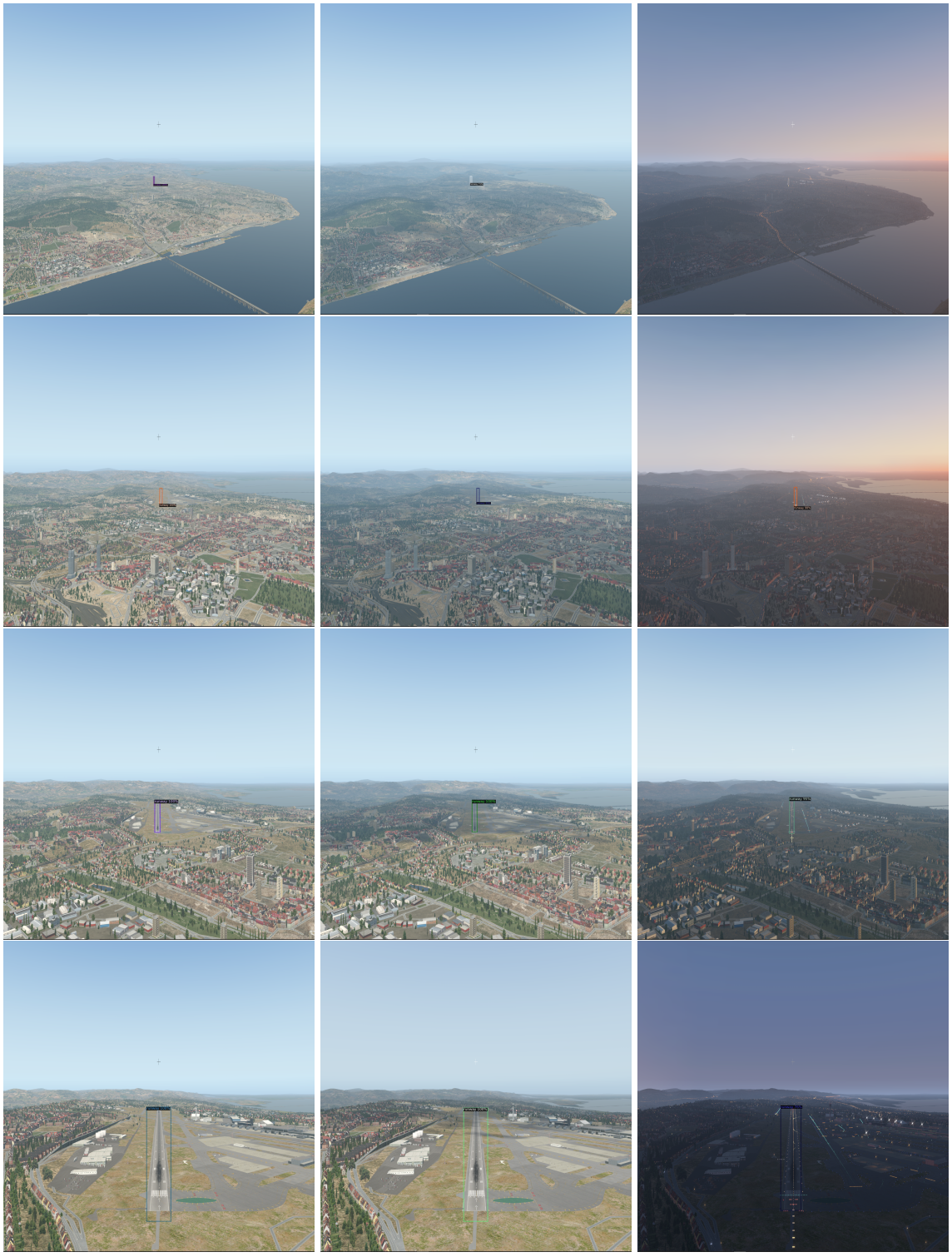


Figure 25: Images of runways during landing approach generated in Xplane. The boxes present in the images are predictions from the object detection model. The columns indicate the environmental condition. The first column is without any environmental condition, the second is with clouds, and the last is with different time of day. Each row indicates an image taken at the same position in the simulator. The distance to the runway for rows 1, 2, 3 and 4 is 12252, 5750, 2707 and 581 metre.

6.5.4 Attitude

This data refers to research question 4. Here three types of attitude variations are applied. Tables 16, 17 and 18 displays the outcome of the evaluation. One Move Expanded and New True in the table refers to the two object detection models described in the Diagram 9.

Model	AP	AP50	AP75	APs	APm	APl
One Move Expanded	55.224	95.482	52.778	45.653	81.321	58.519
New True	59.960	96.403	60.465	50.543	85.582	57.129

Table 16: AP scores from running 2 object detection models on a image set generated by the true method. The image set for this evaluation corresponds to 8000 images at Humberto Delgado Airport, Lisbon, LPPT, with changing pitch applied.

Model	AP	AP50	AP75	APs	APm	APl
One Move Expanded	61.662	98.428	62.509	48.180	82.955	89.772
New True	63.639	97.651	63.359	49.133	88.327	92.691

Table 17: AP scores from running 2 object detection models on a image set generated by the true method. The image set for this evaluation corresponds to 8000 images at Humberto Delgado Airport, Lisbon, LPPT, with changing roll applied.

Model	AP	AP50	AP75	APs	APm	APl
One Move Expanded	55.526	97.424	54.910	43.117	75.427	78.787
New True	58.909	96.787	63.546	48.089	76.543	79.215

Table 18: AP scores from running 2 object detection models on a image set generated by the true method. The image set for this evaluation corresponds to 8000 images at Humberto Delgado Airport, Lisbon, LPPT, with changing yaw applied.

Below, the predictions of the "New True" object detection model is displayed. These predictions are made on images where there varying attitudes. Where the attitudes are yaw, pitch and roll. In the same order, the predictions are visible in the columns of Figure 26.

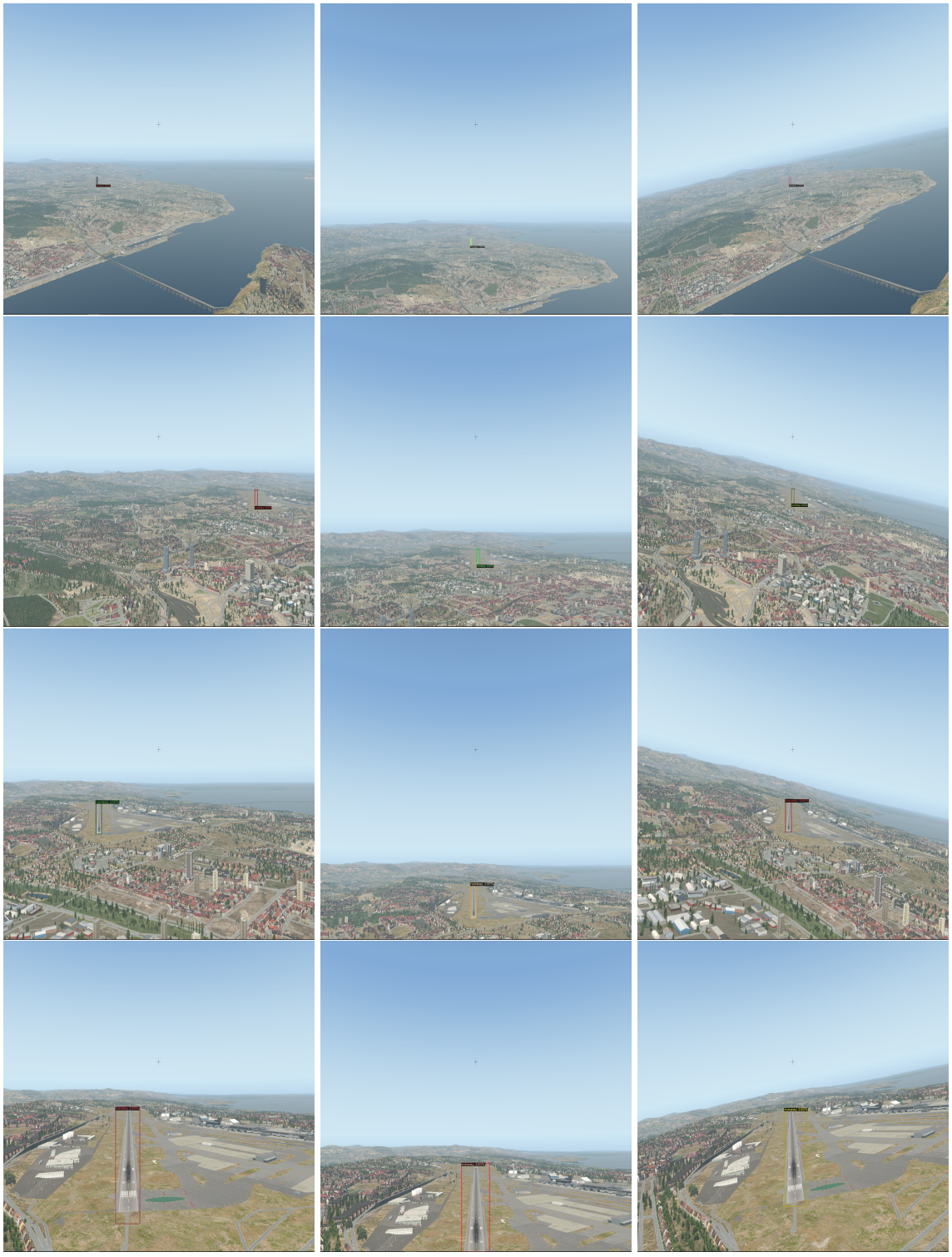


Figure 26: Images of runways during landing approach generated in Xplane. The boxes present in the images are predictions from the object detection model. The columns indicate different attitudes, and rows represent the same image augmented with the attitudes. The first column are changes with yaw, the second is with pitch, and the last is with a different roll. Each row indicates an image taken at the same position in the simulator. The distance to the runway for rows 1, 2, 3 and 4 is 12252, 5750, 2707 and 581 metre.

7. Discussion

In this section, results and limitations are discussed to analyse their significance and relations to each other.

7.1 Flight Data

The flight data that was received from the OpenSky Network allowed for downloading large sets of ADS-B data. Furthermore, the data received from the database contained many parameters necessary for replicating the landing scenarios in a simulated environment. Extended Mode S data could not be established from the OpenSky Network. The Extended Mode S data would have included more parameters that could have allowed for more detailed replication of real flight scenarios. Extended Mode S includes attitude parameters, such as roll and heading [69]. These would allow for replicating a more accurate scenario in the simulator as they would be backed by real-world information.

A problem also identified with using the OpenSky Network was the lack of receivers close to airports. An airport without a receiver prevents ADS-B information from being received when an aircraft is low to the ground as the receiver can not pick up the transmitted signal. This issue was solved by only looking at airports with receivers present. However, this severely prevents many airports around the world to be used. A solution to this would be to use ADS-B information from other databases, as they may be more developed and encompassing in their coverage. However, using another database other than the Opensky Database limits the ability to use the Traffic library.

The Traffic library made it simple in both downloading from the OpenSky database and filtering it. All functions were easy to use and apply. The library was also well documented. The limitations with using the Traffic library were that some functions could not be used. As the access to these tools could not be established. One such tool was the filtering for ADS-B state vector information inside the TMA region. Filtering the flight data for only data positioned in the TMA are used in previous works, [10], [58], [59]. However, the filtering still performed well without access to the TMA region.

METAR data was not added to the flight data. Adding METAR data was not a high priority and thus left behind because it was not necessary for answering the research questions.

7.2 Annotations

The image annotation process was performed fully automatic according to runway and aircraft positions in the simulator. Due to the amounts of datasets needed, this process had to be automatic. 148000 images were generated for 17 different dataset, which would take weeks to annotate manually. This automation comes with some errors regarding the box placement in certain images. In Figure 25 the box is positioned too low, which result in less accurate detection. This error is due to the height positioning of the box, which not accurately represent the height of the runway. Because of this, The height needs to be manually configured to match the runway for as many images as possible. By changing the height of the box and adjusting the altitude values of the runway, this could be achieved. A sample of 50 images at different distances from the dataset was verified to ensure accurate annotation box placement. This configuration was needed every time the runway changed in the simulator. So the impact of this error is only applicable if datasets are compared/trained on different height configurations. For the majority of datasets generated, the same configuration was used. The exceptions are Table 10 which was validated against a new configuration and all changes of runways mentioned in Section 6.5.2. The impact of this error was a decrease of AP across the board, similarly affecting all comparisons.

The annotation algorithm is also limited if the runway is not in the aircraft's field of view. For cases where not the whole runway is seen, the box was unpredictable and, for most cases, faulty. This can be seen in Figure 27. Whether the runway is seen is dependent on the aircraft field of view, altitude, pitch and distance from the runway. To ensure all boxes are positioned correctly, the minimum distance from the runway was set. This can be seen in Figure 28.

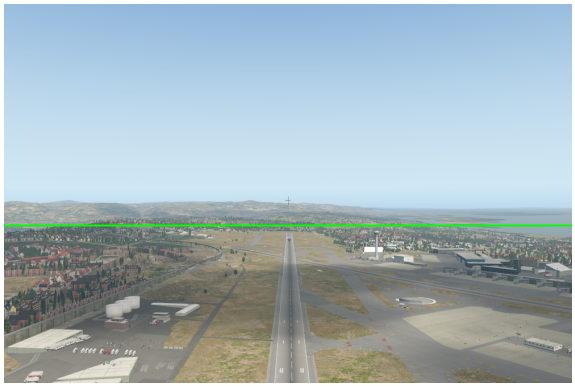


Figure 27: This figure describes a miscalculated annotation box caused by the aircraft position being too close to the runway. Here the annotation box is represented by the green line. The image is captured in the Xplane simulator, where the camera is located by runway 03, at Humberto Delgado Airport, Lisbon, LPPT.

7.3 Research Questions

In this section, direct results from the research questions are discussed. AP metrics answer almost all research questions. These metrics was the most suitable as it incorporates all the calculations needed for determining the prediction accuracy. It is also a well established standard way of making these evaluations.

7.3.1 Methods for Replicating the Distribution of Flight Data

The AP score from Table 9 shows that the model Cube Normal produce the highest AP. That is, the object detection model trained on images generated by positions from the method cube normal. The true model is not included as an official distribution method; it is used as a reference. The cube normal model almost has a 5 point difference from the polyfit and one move models, which is quite significant. As the images used in the evaluation for Table 9 are generated with the true method, the cube normal model therefore performs the best when used on images from real scenarios during landing. Comparing this result to Table 10, which uses the one move method for generating images, the cube normal model also has the best performance in AP. Even compared to the true model, the cube normal outperforms. Here the cube normal model has a 3 point lead from the second-best one move. The interesting thing with the data from Table 10, is that the object detection model for one move has a lower performance than cube normal, even though the images produced for evaluation are generated with the one move method. For the AP in Table 9 the true model has the highest, while in the Table 10, the true model is the third. The reason for this may be due to the images being generated by the true method in the first, which perhaps includes a bias. Thus it may have to be investigated if an object detection model trained on images from one method includes a bias from being evaluated with images generated by the same method. This large reduction in the true models performance from Table 9 to 10 also may be due to overfitting. The true model is trained on a very strict set of flight data with low spread, thus very fit to only one specific type of landing approach scenario. The flight data used in 10 is very spread as it is generated by one move and it contains much flight data. Therefore, this is also a reason for the drastic reduction in the true models' performance.

The reason for the lower AP from Table 10 compared to Table 9, is that the annotation configuration have been changed, which is an error mentioned in Section 7.2.

One noticeable feature present in Table 9 is that for APm and APl, in the one move model, generates higher values than poly and cube normal. While for APs, it is the inverse, Polyfit and Cube normal perform better. This can be translated to one move performs better closer to the runway than the two other methods. APs, APm and APl indicate the AP-score of an object depending on the object's size in the image. Similarly, in Table 10, this feature can also be seen. The reason for this may have to do with the spread of flight data, where higher spread generates

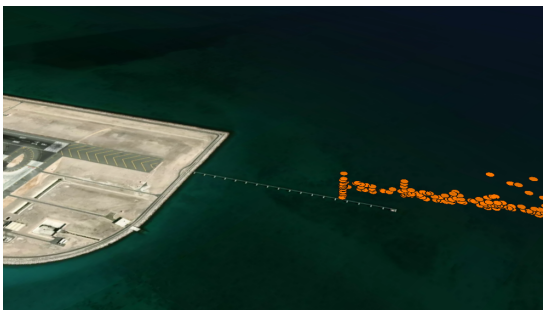
higher AP. Observing the graphs in Figure 22 shows that for the lateral position (first column), one move has a higher spread closer to the runway than poly and cube normal, while inversely at a distance further away. This correlates with the APs, APm and APl scores from Tables 9 and 10, as the scores follow the same behaviour. Due to this feature, combining the one move and cube normal distributions could yield a dataset that has a higher AP across the board.

From the histograms and PDFs present in Figure 19 and 20, it is visible that for two sets of flight data, normal distribution methods do not necessarily resemble real flight data. However, applying a normal distribution still allow for generating new flight data, which still may occur in the real world. It does not necessarily apply to one set of flight data, but it can still occur for others. Using normal distribution for replication of real flight creates more generalised flight data than applying just the real flight data 1-1 in the simulator.

7.3.2 Changing Runways

The images generated for OTHH have an AP of 4.708 for the One Move expanded object detection model and 5.587 for the New True object detection model. Comparing this to the AP in Table 11, which is the same airport the two object detection models were trained on, the results are a 92% respectively 89% reduction in prediction accuracy. Similarly, for the LFPO airport, there is a 49% respectively 45% reduction in prediction accuracy compared to the data from Table 11. To evaluate the factor which causes this reduction, the major changes between the datasets and airport need to be analysed. Firstly, the position of the aircraft is set according to each runways real landing approaches. As seen from Figure 28, the OTHH data have a lower altitude compared to the LPPT and LFPO. This altitude reduction will change the perspective of the runway across the whole dataset and affect the OTHH AP. However, this altitude change is reasonable for the airport due to its location. Landing approaches occur above the water compared to LPPT and LFPO, which occur above land and buildings. Minor errors when positioning the aircraft in the simulator also affect the datasets generated. Setting real latitude and longitude positions in the simulator is not exactly one to one. Thus, when switching between different airports, the error is more noticeable. LPPT was slightly shifted to the left compared to LFPO and OTHH, which can reduce AP.

(a) LPPT

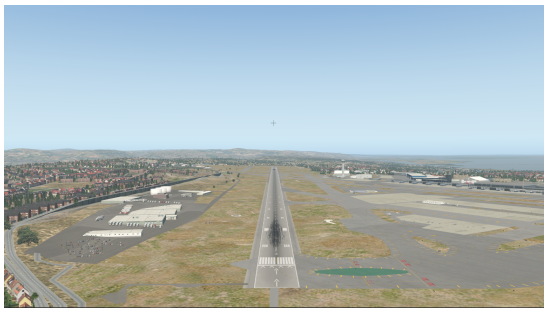


(b) OTHH

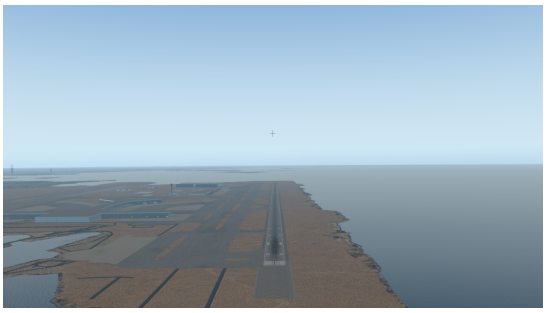
(c) LFPO

Figure 28: These figures show the generated position of the aircraft for the airport LPPT, OTHH and LFPO based on their respective flight data. The orange dots are the generated aircraft positions.

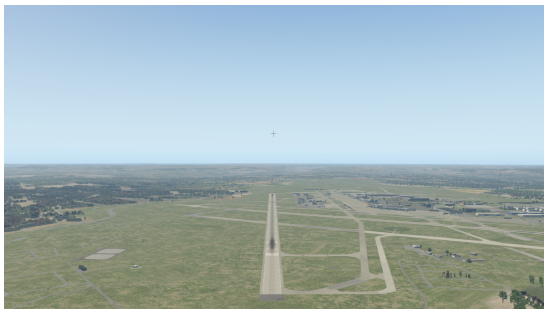
The runways used in LPPT, LFPO and OTHH slightly differ, which can cause AP reductions. As seen in Figures 29, LPPT has a small taxi zone while LFPO has a longer one. Neither LPPT nor LFPO have a blast pad which OTHH have. Since the trained dataset was produced based on LPPT, blast pads are not included and thus result in the redundant prediction, which can be seen in Figure 23c. A similar reduction in AP affected LFPO but with less impact due to the runways similarities. In X-Plane, the OTHH runway also appears much darker compared to the other runways. The darker colour results in the runway being less visible from greater distances which reduce AP. This can be seen in Figure 23a where the runway is barely visible compared to Figure 24a.



(a) LPPT



(b) OTHH



(c) LFPO

Figure 29: These figures show how the runways are displayed in X-Plane for the airports LPPT, OTHH and LFPO. The camera is not positioned equally for all figures.

Since object detection utilises bounding boxes for annotation and detection, small amounts of the environment around the runway are present in each box. The amount of runway environment present is affected by the annotations set during training. Since all runways have a different runway environment in the form of ground colour and taxi zones, it can affect the AP. However, the environment around the runway is not the focus of the CNN to detect. To limit these factors when detecting different runways, segmentation can be used. Segmentation allows for training on the precise position of the runway rather than a box.

Some of the reduction in AP is caused by the annotation configuration. The trained CNN detections would be validated against slightly different annotated boxes, which would yield a reduction in AP. Apart from that, the reductions seem reasonable based on the characteristics of the runways. OTHH is a very different airport compared to LPPT with changes to environment, runway and flight data. The runway environment and runway are quite dark, which makes it hard to detect from a distance. The introduction of the blast pad made it miss predict some images close runway. The altitude for the flight data was considerably lower than LPPT, which changed the runway's perspective. These changes in the environment have a large impact on the detection capabilities of the CNN. LFPO have more similarities with LPPT while considering the environment and runway. Landing approaches also look similar according to the flight data. The large difference between these runways is the environment around the runway. Since the CNN is trained based on only grass colour and pattern from LPPT, the CNN recognises it as a feature. This feature changes from airport to airport and thus result in less AP for LFPO. This could be

solved by using segmentation since it does not include the environment around the runway.

7.3.3 Environmental Conditions:

Tables 14 and 15 display the AP scores of evaluating images with clouds and time changes present. The object detection models are the same described in Section 7.3.2. The reduction in prediction accuracy is 24% respectively 18% from applying clouds to the image sets. This reduction is calculated by comparing it to the data from Table 11, with no environmental conditions. Time of day changes applies a 14% and 12% reduction. By observing the differences with images containing no environmental conditions, clouds and changing time in Figure 25, the reason for this decrease can be analysed.

Applying clouds to the environment adds variation to how the runway looks. If there are clouds present, the runway can have some darker spots that cover the whole runway or just some dark spots. The object detection has issues with detecting the runway because of this, as it is not used to this variation. The object detection is used to clear runways and no variation of light, as those visible in the first column Figure 25. However, in Table 14, the APs received a more significant reduction compared to APm and APl. This reduction means that the variations of dark spots on the runway have a considerable impact when the aircraft is far from the runway. This is similar, but a less extreme case of the effect OTHHs runway had on the object detection in section 7.3.2.

The time of day environmental conditions adds shadows from buildings and a hue from the setting/rising sun. A 14% and 12% reduction in the object detection models ability to identify and localise the runway during the time of day change is not major. Although images are not generated at night conditions, it should not be a major reduction in prediction accuracy. Likewise, from clouds effects, the reduction of APs is slightly more compared to APm and APl in Table 15

Clouds affect object detection more than the time of day. The time of day images in Figure 25 adds many variations to the images. Although, those used in the figure are on the more extreme for display purposes. The majority of images from this dataset will be similar to the real dataset used for evaluation in Table 11. Because of that, the extreme cases displayed in Figure 25 might affect the detection more compared to equal amounts of cloud augmented images.

Rain was supposed to be implemented in the environmental conditions, however when this was generated in the simulator the runway failed to load in. This failure to load may be due to it taking too long to load from generating water effects on the ground. The result from this is visible in Figure 30.

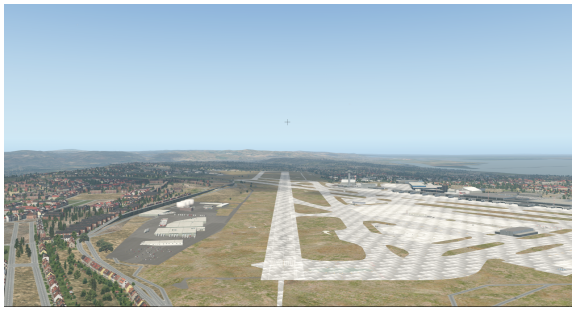


Figure 30: Image of the runway at LPPT with rain effects applied in Xplane. The white texture on the runway and taxiways indicates that they have failed in rendering.

In contrast to the result yielded here with applying environmental conditions, Levin and Vidimlic [61] also apply weather effects, and they achieve an 18.1% decrease. This is somewhat similar to the result presented here. However, the setup and configuration are different, and hence, the comparison is not fair.

7.3.4 Attitude

Pitch applied on images cause an 8% and 5% reduction in prediction accuracy. Roll has a positive effect on prediction accuracy, increasing it by 2% and 1% respectively. Lastly, yaw yields an 8% and

6% reduction. These attitude parameters are compared to the results with no changes to attitude in Table 11. Tables 16, 17 and 18 show the AP received from evaluating attitude variations. The images present in Figure 26 show how these variations to attitude can look. All that changes with pitch and yaw are where the runway is located in the images. Roll rotates the images, which also rotates the runway. Not all these are realistic attitude angles. The intervals the attitude angles are set at can occur in the entire span of a flight trajectory. This causes some unrealistic images, for example, a major roll occurring close to the runway.

For attitude changes, the only significantly affected AP metric is where the images are close to the runway, i.e. the APl. APm is also affected, although it is not major. For pitch, the prediction accuracy decreases by about 38% for the APl metric. This is a major reduction in the CNNs (object detection models) ability to detect the runway at close range. One possible reason for this decrease is due to the runway being cut off in the image if the pitch is high and the aircraft is close to the runway. As the object detection models are not trained on portions of the runway, it may have a difficulty predicting it. An example of this is visible in Figure 26, final row middle column.

Roll yields a positive increase in AP. The increase is small; therefore, it may just be coincidental. Nevertheless, there is no reduction, which is surprising since the effect on the images is significant. However, there maybe exist some kind of rotation transformation built-in Detectron2 for training, which negates this reduction produced by the roll. Although, from some analysis, this should not be the case. A more rigorous investigation needs to be performed for drawing any conclusions. The most affected prediction accuracy in the roll image set is the reduction in APl. Although, this is not a significant difference. For roll, the reduction in APl is 5% for one move expanded and 1% for new true.

The ability for the object detection to predict when images of yaw are applied yields a 16% reduction when close to the runway (APl). Even when at distances further from the runway, the APm decreases by about 11%.

For variations in attitude, considerations have to be made when generating images closer to the runway. Both pitch and yaw significantly affected the object detection when the runway was close.

7.4 Reliable Dataset Generation

Based on the effect of variations in the environment discussed in Section 7.3, their impact needs to be considered while generating a reliable dataset.

The distribution used for generating datasets did not receive better AP compared to the dataset simulated with real flight data in Table 9. However, cube normal and one move had better AP than real data when validated against the one move distribution in Table 10. This supports the claim that the distribution is better at detecting specific cases during landing scenarios. This feature needs to be included to detect more specific cases and is also discussed in a public report written by EASA [1] about synthesised generated data.

The runway appearance, runway environment, and flight data had a large impact on AP from the change of runway analysis. Different Runway appearances need to be considered in the training due to its impact on reducing detection capabilities at long distances. Runway environment should also be included either by training for images with a different environment or using segmentation. Finally, to ensure detection is correct on different airports, the dataset needs to cover landings at different altitudes and descent angles. As discussed in Section 7.3.2, the flight path is lower compared to the other runways, which change its perspective. This could be solved using a broad, comprehensive distribution to position the aircraft during the generation of images.

The environmental conditions analysed mostly affected the distant detection of the CNN. This was due to their ability to make the runway darker, thus making it harder to detect. For a reliable dataset, this needs to be included to allow aircraft to operate during the whole day and under cloudy conditions.

Attitude greatly change based on aircraft and weather condition present. Their effect on the detection capabilities mainly was presented close to medium range from the runway. Pitch had a large impact when close to the runway due to its ability to move the runway out of camera vision. This problem relates more to camera position and alignment than the generation of images. Yaw had an impact on medium to close range AP due to the change in perspective. These variations

are similar to typical image augmentation techniques, which should be included in the training of a reliable CNN.

7.5 Validity

One validity concern is that the flight data does not represent real landing approaches. This may occur from the filtering of flight data and placement of the data relative to the airport.

As the placement of data relative to the runway is dependant on points and bearings that represent the runway, a systematic error may be present if these are placed wrong. An example of this is visible in Figure 11. In the example, the point "Runway Start LPPT" is used for placing the ADS-B data with respect to the runway. The point is necessary for calculating the parameters of lateral positioning and aircraft to the runway. As the point is not placed on the centerline, it is slightly to the left; the flight data will have a systematic error. This causes the flight data to be placed more to the left than what the real aircraft was. This error could also affect altitude, which would result in misrepresented runway perception compared to real scenarios. This can be resolved by placing the runway/centerline coordinate with maps and other databases such as SkyVector, without using a database in Traffic that has this information. Alternatively, the deviation from the true centerline can be compensated for by adding or subtracting the real deviation, discovered by using maps and calculating the distance. This deviation is minor and does not pose a major issue with replicating real flight data, at least for the airports this paper investigated. At close distances, this causes the most effect on the images. If this method of calculating the flight data relative to the runway is performed more on different airports, it is necessary to keep track of this deviation. If this deviation becomes too big, it can become an issue.

One possible validity concern may be that the size of the images sets used is not enough to generate a conclusive description of the runways features. However, the image sets used are substantial compared to similar implementations demonstrated in [61]. Furthermore, the images are not introduced to several different kinds of variations; thus, many images are not required.

Invalid or erroneous points are removed during the filtering for the landing approach made in Traffic. This may cause an issue if a point is valid and then removed. If this occurs, sections or parts of flight data will not be represented in the images and thus not trained on during the object detection. However, by observing the flight data before and after removing points, any valid points which would be removed should have been noticed. Furthermore, gathering more flight data also limit the effects of removing these small amounts of valid points during filtering.

For the polyfit method, a third-order function was used in standard deviation for altitude. However, in hindsight, a constant value would have been a better solution.

8. Conclusions

This paper aimed to determine what effects variations in the environment have on prediction accuracy on object detection/CNNs, focusing on aviation domain application, i.e. runway detection during landing approach. The methods used to achieve this was firstly a real flight data gathering and filtering step. Secondly, applying this flight data in a simulated environment to generate datasets that resemble aircraft landings occurring in real life. This can then be annotated to train a CNN for runway detection. By adding variations in different datasets, the resulting images are affected and can be evaluated against each other using prediction accuracy. The variations considered in this thesis are synthesised data generated based on real data, weather conditions, different runways and aircraft attitude.

8.1 Research Question 1

Research question 1 is the following "**What method of generating distributions from real data produces the highest prediction accuracy from generated images?**"

By gathering real landing approach flight data from the LPPT airport, their behaviours could be analysed. Based on this analysis, characteristics of these landings were utilised to create three sampling distributions that augmented the flight data and generated datasets in the simulated environment X-Plane. By training CNNs on these dataset and comparing them, their effects on detection capabilities can be evaluated. From this comparison, the cube normal distribution had the best performance according to AP among the distributions. Cube normal had 63.2296% AP, which is approximately 5% more than the other distributions when validated against real flight data, i.e. those from ADS-B. When validated against the one move distribution, cube normal is approximately 3.5 points higher with a total of 47.185 AP. However, when considering AP at different ranges, e.g. APs, APm and APl, the results differ. One move has better detection capabilities while close to the runway according to Apl and Apm. Based on this, a combination of cube normal and one move would yield a better distribution overall.

8.2 Research Question 2

Research question 2 is the following "**How does changing airport runways affect the prediction accuracy of the object detection?**"

To evaluate the impact of different runways, two validation datasets were created based on the real flight data of the airports OTHH and LFPO. These sets were used as validation against two trained CNNs based on real flight data and one move distribution on the LPPT airport. The validation process concludes a large reduction in AP for both OTHH and LFPO. OTHH have 5.587% AP compared to real data from LPPT and 4.708% AP compared to the one move distribution. LFPO have 30.294% AP compared to real data from LPPT and 23.208% AP compared to the one move distribution. LFPO is more similar to LPPT compared to OTHH, which made that reduction much less substantial. Based on analysing the changes between these airports, factors such as runway appearance, runway environment and flight data positioning had a large impact on the detection capabilities of the CNN. All of these variables were different on OTHH, and the runway environment was different on LFPO. Due to these reductions in AP, changing the airport runway significantly impacts the CNN detection capabilities.

8.3 Research Question 3

Research question 3 is the following "**How does environmental conditions affect prediction accuracy of the object detection? **"

The environmental conditions evaluated consisted of changes to time and a varying amount of clouds. Images with clouds and changing times were generated. These image sets were evaluated against two object detection models that had not been trained on these conditions. Another image set was also used during evaluation, which had no environmental conditions for comparison.

The AP decreased from 60.242 and 62.877 down to 45.493 and 51.763 for cloudy conditions. This was a reduction of 24% from one object detection model and 18% from the other. For the changing time of day, the AP went down to 51.853 and 55.062; this is also compared against 60.242

and 62.877 from no variations. Applying changing time of day yielded a 14% and 12% decrease in an object detection's ability to predict the location of a runway. The time of day does not yield a major reduction; however, the images for the time of day changes are not generated at night conditions. There are also many normal images in time variations, i.e. those the object detections are trained on. Both variations did make the runway darker, which like the darker runway, affected distant detection more.

Variations in environmental conditions are, therefore, an important feature to consider when training an object detection for detecting a runway.

8.4 Research Question 4

Research question 4 is the following "**How does aircraft attitude affect prediction accuracy of the object detection?**"

The attitude composes of yaw, pitch and roll. Images were generated where each of these parameters varied. These image sets were then evaluated with two object detection models trained without any attitude variations. An image set with no attitude variations was also generated. This allows for determining the effects these attitudes have.

The result of the evaluation consists of AP metrics. The AP is calculated with two trained object detection models. For pitch, the AP reduced by 8% and 5%. Roll increased the AP by 2% and 1%. Lastly, yaw reduced the AP by 8% and 6%. However, the AP was mostly affected at images where the runway was closest.

Instead, looking at AP where the runway is close yields the following: For pitch, the average prediction accuracy from two object detection models reduces by about 38%. Yaw reduce about 16% and lastly the roll decrease with 1% and 5%. Because of this, for creating a reliable object detection model, both pitch and yaw at close range to runway have to be given special consideration during training.

8.5 Research Question 5

Research question 5 is the following "**What variables need to be considered when generating reliable datasets used for safety-critical landing approaches?**"

Based on the results and analysis presented for RQ1, RQ2, RQ3 and RQ4, all features/variables tested are important but affect detection in different ways. For a reliable dataset used for safety-critical landing approaches, all variables need to be included. But not at all distances from the runway. For example, runway appearance and attitude only affect close to medium range from the runway while weather affects long-range detection. Positioning distributions also need to be considered since they improve the detection of special landing scenarios. They would also make the dataset more transferable to other airport runways due to the variations in flight data.

8.6 Future Work

Extended Mode S flight data could be added. Such that the attitude angles used are more representative of real flights. This information can be received, although it may be limited in amount. Olive and Bieber have had success in receiving this information [69].

Implementing METAR data to the flight data could also expand the detail necessary for a realistic representation. This weather data could be implemented by querying METAR databases with a timestamp and landing airport from ADS-B data. The METAR data addition would perhaps allow for filtering flight data which is more extreme. If there is, for example, much wind, the flight is induced to drift. This more extreme flight data could then be used in the simulator. Replicating the real weather from METAR data in the simulator would allow for very realistic landing scenarios which an aircraft might observe in real life.

Instead of using object detection, applying semantic segmentation is an interesting alternative. Semantic segmentation would allow for prediction and annotation of only the runway, excluding the terrain captured with applying a box limited by object detection. This alternative may allow for higher prediction accuracy during changing runways or airports, as the terrain is not included in the annotation and prediction.

Many more types of variations could be investigated by the methods applied in this paper. Interesting variations may include objects present on the runway, snow, rain, brightness, motion blur and many more. Different combination of variations could also be analysed to see their effect in conjunction with each other. Other features which may be necessary to evaluate is how important changing altitude versus lateral position is. The setup for this may include only changing the lateral position and setting altitude so that it always is the same or the other way round.

A clear step ahead is to also train on these variations for seeing how a neural network will react in its predictions. For example, how much would training on the environmental conditions allow for increasing the prediction accuracy?

According to EASA [1], the discrepancy between applying synthetic data for a target in the real world need to be analysed. This is also a challenge which needs to be undertaken, if the work in this thesis is to be used for real-life applications. This is also dependant on how realistic the simulator looks. In this case, Xplane is somewhat realistic. However, there may be those that have a higher resemblance to real environments. A possible future work can therefore be to investigate how the prediction accuracy changes depending on the simulation environment.

References

[1] J. M. Cluzeau, X. Henriquel, G. Rebender, G. Soudain, L. Dijk, A. Gronskiy, D. Haber, C. Perret-Gentil and R. Polak, *Concepts of Design Assurance for Neural Networks (CoDANN)*. 2020, [Accessed May. 18, 2021]. [Online]. Available: https://www.easa.europa.eu/sites/default/files/dfu/EASA-DDLN-Concepts-of-Design-Assurance-for-Neural-Networks-CoDANN.pdf.

[2] FAA, *Airplane flying handbook*, [Accessed May. 18, 2021], Federal Aviation Administration, 2016. [Online]. Available: https://www.faa.gov/regulations_policies/handbooks_manuals/aviation/airplane_handbook/.

[3] *Airport traffic patterns*, [Accessed May. 18, 2021], Federal Aviation Administration, 2016. [Online]. Available: https://www.faa.gov/regulations_policies/handbooks_manuals/aviation/airplane_handbook/media/09_afh_ch7.pdf.

[4] *Aeronautical information manual - aim*, [Accessed May. 18, 2021]. [Online]. Available: https://www.faa.gov/air_traffic/publications/atpubs/aim_html/chap2_section_3.html.

[5] B. Syd Ali, W. Ochieng, A. Majumdar, W. Schuster and T. Chiew, 'Ads-b system failure modes and models,' *Journal of Navigation*, vol. 67, pp. 1–23, Nov. 2014.

[6] *How nextgen works*, [Accessed May. 18, 2021], 2020. [Online]. Available: https://www.faa.gov/nextgen/how_nextgen_works/.

[7] *The opensky network api documentation — the opensky network api 1.4.0 documentation*, [Accessed May. 18, 2021]. [Online]. Available: https://opensky-network.org/apidoc/index.html.

[8] *What is nextgen?* [Accessed May. 18, 2021], 2021. [Online]. Available: https://www.faa.gov/nextgen/what_is_nextgen/.

[9] S. Peeters and G. Guastalla, *Analysis of vertical flight efficiency during climb and descent*, 2017.

[10] A. Lemetti, T. Polishchuk, V. Polishchuk, R. S. García and X. Menéndez, 'Identification of significant impact factors on arrival flight efficiency within tma,' 2020.

[11] I. Petchenik, *Understanding extended mode s data in flightradar24 | flightradar24 blog*, [Accessed May. 18, 2021], 2016. [Online]. Available: https://www.flightradar24.com/blog/understanding-extended-mode-s-data-in-flightradar24/.

[12] *Metar api documentation | checkwx aviation weather api*, [Accessed May. 18, 2021]. [Online]. Available: https://www.checkwxapi.com/documentation/metar.

[13] M. Schäfer, *Historical database*, [Accessed May. 18, 2021], 2020. [Online]. Available: https://opensky-network.org/data/impala.

[14] [Accessed May. 18, 2021], 2021. [Online]. Available: https://opensky-network.org/about/about-us.

[15] *Traffic – air traffic data processing in python — traffic documentation*, [Accessed May. 18, 2021]. [Online]. Available: https://traffic-viz.github.io/index.html.

[16] X. Olive, 'Traffic, a toolbox for processing and analysing air traffic data,' *Journal of Open Source Software*, vol. 4, 2019, ISSN: 2475-9066.

[17] J. Brooks-Bartlett, *Probability concepts explained: Probability distributions (introduction part 3)*, [Accessed May. 18, 2021], 2018. [Online]. Available: https://towardsdatascience.com/probability-concepts-explained-probability-distributions-introduction-part-3-4a5db81858dc.

[18] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006, pp. 24, 78–80.

[19] C. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006, pp. 4–11.

[20] S. I. Group, *Aerospace recommended practice*, [Accessed May. 18, 2021], SAE Aerospace, 2010 12. [Online]. Available: https://www.sae.org/standards/content/arp4754a/.

[21] *X-plane*, [Accessed May. 18, 2021], Laminar Research, 2021. [Online]. Available: https://www.x-plane.com/.

[22] X-Plane, *X-plane 11 desktop manual*, [Accessed Jan. 23, 2021], Nov. 2020. [Online]. Available: https://x-plane.com/manuals/desktop/.

[23] D. J. Fremont, E. Kim, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli and S. A. Seshia, *Scenic's documentation*, Version 2 [Accessed May. 18, 2021], Dec. 2020. [Online]. Available: https://scenic-lang.readthedocs.io/en/latest/.

[24] *Carla*, [Accessed May. 18, 2021], 2021. [Online]. Available: https://carla.org/.

[25] *Gta*, [Accessed May. 18, 2021], 2021. [Online]. Available: https://www.rockstargames.com/games/V.

[26] *Lgsvl*, [Accessed May. 18, 2021], 2021. [Online]. Available: https://www.lgsvlsimulator.com/.

[27] *Webots*, [Accessed May. 18, 2021], 2021. [Online]. Available: https://cyberbotics.com/.

[28] *Xplaneconnect*, Nasa Version 1.2.1[Accessed May. 18, 2021], May 2017. [Online]. Available: https://github.com/nasa/XPlaneConnect.

[29] *Datarefs for x-plane 1041*, [Accessed May. 18, 2021],[compiled Mon Oct 5 23:29:08 2015]. [Online]. Available: http://www.xsquawkbox.net/xpsdk/docs/DataRefs.html.

[30] *Xpc client reference*, [Accessed May. 18, 2021]. [Online]. Available: https://github.com/nasa/XPlaneConnect/wiki/XPC-Client-Reference.

[31] *Xplmgraphics api*, [Accessed May. 18, 2021]. [Online]. Available: https://developer.x-plane.com/sdk/XPLMGraphics/.

[32] M. T. Hagan, H. B. Demuth, M. H. Beale and O. D. Jesús, *Neural Network Design*, 2nd ed. 2014, [Accessed May. 18, 2021].

[33] M. Nielsen, *How the backpropagation algorithm works*, [Accessed May. 18, 2021], Dec. 2019. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap2.html.

[34] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016, [Accessed May. 18, 2021].

[35] M. Nielsen, *Using neural nets to recognize handwritten digits*, [Accessed May. 18, 2021], Dec. 2019. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap1.html.

[36] A. M. Mateusz Buda and M. A. Mazurowski, *A systematic study of the classimbalance problem in convolutional neural networks*, arXiv:1710.05381, 2018.

[37] R. Hecht-Nielsen, *Theory of the backpropagation neural network*, 1988.

[38] J. Gua, Z. Wangb, J. Kuenb, L. Mab, A. Shahroudyb, B. Shuaib, TingLiub, X. Wangb, L. Wangb, G. Wangb, J. Caic and T. Chen, *Recent advances in convolutional neural networks*, arXiv:1512.07108, Sep. 2017.

[39] S. Saha, 'A comprehensive guide to convolutional neural networks — the eli5 way,' *towards data science*, Dec. 2018.

[40] S. Y. Min Lin Qiang Chen, *Network in network*, arXiv:1312.4400, Mar. 2014.

[41] Z.-Q. Zhao, P. Zheng, S.-t. Xu and X. Wu, *Object detection with deep learning: A review*, arXiv:1807.05511v2, Apr. 2019.

[42] C. Szegedy, A. Toshev and D. Erhan, *Deep neural networks for object detection*, 2013.

[43] A. Gonfalonieri, 'How to build a data set for your machine learning project,' Feb. 2019, [Accessed May. 18, 2021]. [Online]. Available: https://towardsdatascience.com/how-to-build-a-data-set-for-your-machine-learning-project-5b3b871881ac.

[44] G. V. Horn, O. M. Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona and S. Belongie, *The inaturalist species classification and detection dataset*, arXiv:1707.06642, Jun. 2017.

[45] P. Lambert, 'An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets,' Universiti Teknologi MARA, Selangor,Malaysia, Tech. Rep., Dec. 2013, [Accessed May. 18, 2021]. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-4585-18-7_2.

[46] N. J. Taeho Jo profile, *Class imbalances versus small disjuncts*, ACM, Jun. 2004.

[47] H. L. V. Hongyu Guo, *Learning from imbalanced data sets with boosting and data generation: Thedataboost-im approach*, Research Gate, Jun. 2004.

[48] S. M. Miroslav Kubat, *Addressing the curse of imbalanced training sets: One-sided selection*, Research Gate, Jun. 2000.

[49] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo and R. Girshick, *Detectron2*, https://github.com/facebookresearch/detectron2, [Accessed May. 18, 2021], 2019.

[50] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo and R. Girshick, *Detectron2: A pytorch-based modular object detection library*, [Accessed May. 18, 2021], 2019. [Online]. Available: https://ai.facebook.com/blog/-detectron2-a-pytorch-based-modular-object-detection-library-/.

[51] F. Inc, *Pytorch*, https://github.com/pytorch/pytorch, [Accessed May. 18, 2021].

[52] T. Shah, *Measuring object detection models—map—what is mean average precision?* [Accessed May. 18, 2021], 2018. [Online]. Available: https://towardsdatascience.com/what-is-map-understanding-the-statistic-of-choice-for-comparing-object-detection-models-1ea4f67a9dbd.

[53] R. J. Tan, *Breaking down mean average precision (map)*, [Accessed May. 18, 2021], 2019. [Online]. Available: https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52.

[54] J. Hui, *Map (mean average precision) for object detection*, [Accessed May. 18, 2021], 2018. [Online]. Available: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173.

[55] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick and P. Dollár, *Microsoft coco: Common objects in context*, [Accessed May. 18, 2021], 2015. arXiv: 1405.0312 [cs.CV].

[56] *Coco - common objects in context*, [Accessed May. 18, 2021]. [Online]. Available: https://cocodataset.org/#detection-eval.

[57] M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic and M. Wilhelm, 'Bringing up opensky: A large-scale ads-b sensor network for research,' in *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, 2014, pp. 83–94.

[58] S. Lucie, 'Evaluation of arrival sequencing at arlanda airport,' M.S. thesis, Linköping University, The Institute of Technology, 2020.

[59] A. Lemetti, T. Polishchuk, R. Sáez and X. Prats, 'Evaluation of flight efficiency for stockholm arlanda airport arrivals,' in *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, 2019, pp. 1–8.

[60] P. Kamsing, P. Torteeka, S. Yooyen, S. Yenpiem, D. Delahaye, P. Notry, T. Phisannupawong and S. Channumsin, 'Aircraft trajectory recognition via statistical analysis clustering for suvarnabhumi international airport,' in *2020 22nd International Conference on Advanced Communication Technology (ICACT)*, 2020, pp. 290–297.

[61] A. Levin and N. Vidimlic, 'Improving situational awareness in aviation: Robust vision-based detection of hazardous objects,' M.S. thesis, Mälardalen University, School of Innovation, Design and Engineering, 2020.

[62] B. Ajith, S. D. Adlinge, S. Dinesh, U. P. Rajeev and E. S. Padmakumar, 'Robust method to detect and track the runway during aircraft landing using colour segmentation and runway features,' in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, 2019, pp. 751–757.

[63] X. Wang, B. Li and Q. Geng, 'Runway detection and tracking for unmanned aerial vehicle based on an improved canny edge detection algorithm,' in *2012 4th International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 2, 2012, pp. 149–152.

[64] J. Akbar, M. Shahzad, M. I. Malik, A. Ul-Hasan and F. Shafait, 'Runway detection and localization in aerial images using deep learning,' in *s2019 Digital Image Computing: Techniques and Applications (DICTA)*, 2019, pp. 1–8.

[65] X. Olive, *Impact of covid-19 on worldwide aviation*, [Accessed May. 18, 2021]. [Online]. Available: https://traffic-viz.github.io/scenarios/covid19.html.

[66] X. Olive and P. Bieber, *Quantitative assessments of runway excursion precursors using mode s data — traffic documentation*, 2018. [Online]. Available: https://traffic-viz.github.io/paper/runway_excursion.html.

[67] T. Dreoss, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte and S. A. Seshia., *Verifai*, Version 2.0.0b1[Accessed May. 18, 2021], Oct. 2020. [Online]. Available: https://github.com/BerkeleyLearnVerify/VerifAI.

[68] *A320 normal procedures*, Section: LANDING[Accessed April. 17, 2021], Nov. 2020. [Online]. Available: https://www.theairlinepilots.com/forumarchive/a320/a320-normal-procedures.pdf.

[69] X. Olive and P. Bieber, 'Quantitative assessments of runway excursion precursors using mode s data,' Jun. 2018.

A X-Plane Settings

The setting used during the generation of images in X-Plane can be seen in Figure 31. These settings were used to include as much visual detail as possible for a reasonably low loading time. The settings affect how fast the scene can be loaded in, which affects the time needed to generate each image. Number of world objects was not maxed out due to its function to spawn more aircraft in the air. This function, caused in some cases, an aircraft to spawn in front of the camera, which could block the vision of the runway. Reflection detail was also not maxed out due to reflection graphical bugs while generating certain cases.

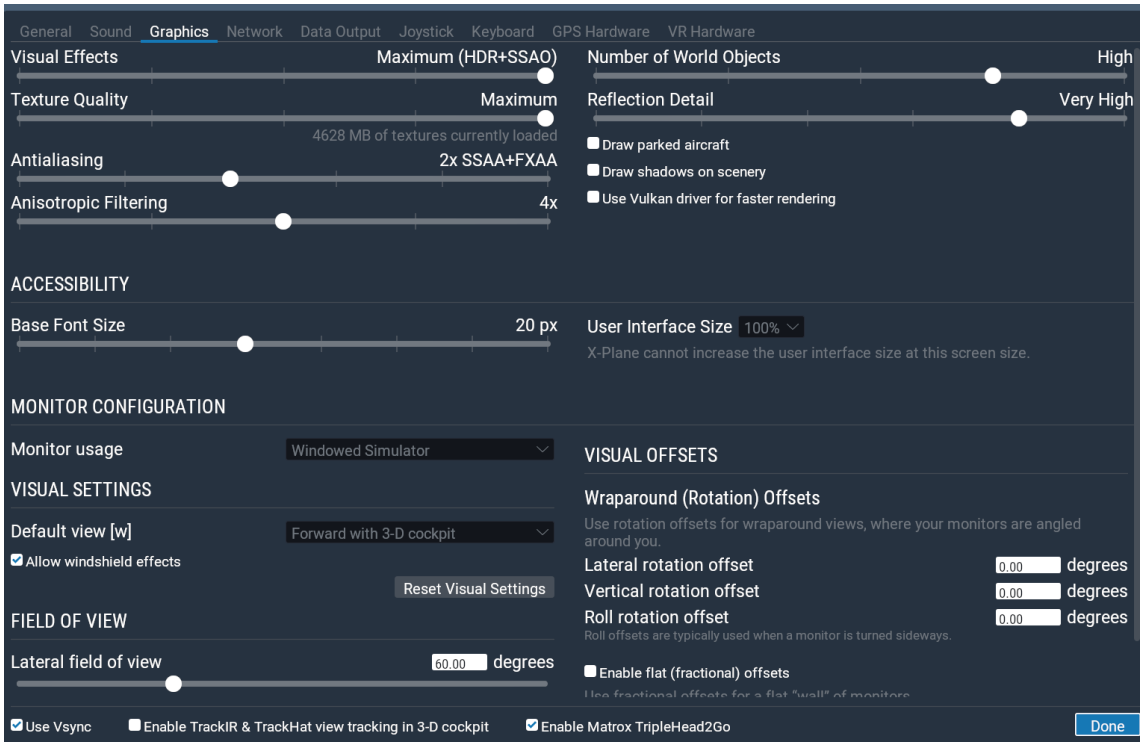


Figure 31: This figure displays the setting used for the X-Plane simulator during the generation of datasets.